Weighted Vote-Based Classifier Ensemble for Named Entity Recognition: A Genetic Algorithm-Based Approach

ASIF EKBAL and SRIPARNA SAHA, Indian Institute of Technology

In this article, we report the search capability of Genetic Algorithm (GA) to construct a weighted vote-based classifier ensemble for Named Entity Recognition (NER). Our underlying assumption is that the reliability of predictions of each classifier differs among the various named entity (NE) classes. Thus, it is necessary to quantify the amount of voting of a particular classifier for a particular output class. Here, an attempt is made to determine the appropriate weights of voting for each class in each classifier using GA. The proposed technique is evaluated for four leading Indian languages, namely Bengali, Hindi, Telugu, and Oriya, which are all resource-poor in nature. Evaluation results yield the recall, precision and F-measure values of 92.08%, 92.22%, and 92.15%, respectively for Bengali; 96.07%, 88.63%, and 92.20%, respectively for Hindi; 78.82%, 91.26%, and 84.59%, respectively for Telugu; and 88.56%, 89.98%, and 89.26%, respectively for Oriya. Finally, we evaluate our proposed approach with the benchmark dataset of CoNLL-2003 shared task that yields the overall recall, precision, and *F*-measure values of 88.72%, 88.64%, and 88.68%, respectively. Results also show that the vote based classifier ensemble identified by the GA-based approach outperforms all the individual classifiers, three conventional baseline ensembles, and some other existing ensemble techniques. In a part of the article, we formulate the problem of feature selection in any classifier under the single objective optimization framework and show that our proposed classifier ensemble attains superior performance to it.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Named entity recognition*

General Terms: Algorithms, Design, Experimentation

Additional Key Words and Phrases: Language independent named entity recognition, genetic algorithm, maximum entropy, conditional random field, support vector machine, classifier ensemble, feature selection

ACM Reference Format:
Ekbal, A. and Saha, S. 2011. Weighted vote-based classifier ensemble for named entity recognition: A genetic algorithm-based approach. ACM Trans. Asian Lang. Inform. Process. 10, 2, Article 9 (June 2011), 37 pages.
DOI = 10.1145/1967293.1967296 http://doi.acm.org/10.1145/1967293.1967296

1. INTRODUCTION

Named Entity Recognition (NER) is a well-established task that has immense importance in many Natural Language Processing (NLP) application areas such as Information Retrieval [Mandl and Womser-Hacker 2005], Information Extraction [Pasca et al. 2006], Machine Translation [Babych and Hartley 2003], Question Answering [Pizzato

The authors equally contributed to this work.

Author's address: A. Ekbal and S. Saha, Department of Computer Science and Engineering, Indian Institute of Technology, Patna, India, 800013; e-mail: {asif, sriparna}@iitp.ac.in.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org.
© 2011 ACM 1530-0226/2011/06-ART9 $10.00
DOI 10.1145/1967293.1967296 http://doi.acm.org/10.1145/1967293.1967296

et al. 2006], and Automatic Summarization [Nobata et al. 2002] etc. The objective of NER is to identify and classify every word/term in a document into some predefined categories such as name, location name, organization name, miscellaneous name (date, time, percentage, and monetary expressions, etc.), and none-of-the-above.

The main approaches to NER can be grouped into three main categories, namely rule-based, machine learning-based, and hybrid approach. Rule-based approaches focus on extracting names using a number of handcrafted rules. Generally, these systems [Aone et al. 1998; Humphreys et al. 1998; Mikheev et al. 1998, 1999] consist of a set of patterns using grammatical (e.g., part of speech), syntactic (e.g., word precedence), and orthographic features (e.g., capitalization) in combination with dictionaries. These kinds of systems have better results for restricted domains and are capable of detecting complex entities that are difficult with learning models. However, rule-based systems lack portability and robustness, and furthermore the high cost of maintenance of rules increases even when the data is slightly changed. These types of systems are often domain dependent, language specific, and do not necessarily adapt well to new domains and languages.

Researchers nowadays are popularly using machine-learning approaches for NER because these are easily trainable, adaptable to different domains and languages, and their maintenance is also less expensive. The ML techniques can be grouped into the following three categories: supervised, semi-supervised, and unsupervised. The idea of supervised learning is to study the features of positive and negative example of NE over a large collection of annotated documents and design rules that capture instances of a given type. Some of the representative supervised machine learning approaches used in NER are Hidden Markov Model (HMM) [Bikel et al. 1999; Miller et al. 1998], Maximum Entropy (ME) [Borthwick 1999; Borthwick et al. 1998], Decision Tree [Bennet et al. 1997; Sekine 1998], and Conditional Random Field (CRF) [Lafferty et al. 2001; McCallum and Li 2003]. The main shortcoming of supervised learning is the requirement of a large annotated corpus, which is very difficult to obtain for many resource-constrained languages. The unavailability of such resources and the prohibitive cost of creating them lead to two alternative learning methods: semi-supervised and unsupervised. The term semi-supervised (or, weakly supervised) is relatively recent. The main technique for semi-supervised approach [Collins and Singer 1999; Riloff and Jones 1999; Yangarber et al. 2002] is bootstrapping and involves a small degree of supervision, such as a set of seeds, for starting the learning process. The typical approach in unsupervised learning is clustering. For example, one can try to gather NEs from clustered groups based on the similarity of context. There are other unsupervised methods too. Basically, the techniques rely on lexical resources (e.g., WordNet), on lexical patterns and on statistics computed on a large unannotated corpus. The typical unsupervised systems are Alfonseca and Manandhar [1999], Shinyama and Sekine [2004], and Etzioni et al. [2005]. In hybrid systems [Mikheev et al. 1998; Srihari et al. 2002; Yu 2007], the goal is to combine rule-based and machine learning-based methods, and develop new methods using strongest points from each one. Although hybrid approaches can get better results than some other approaches weakness of handcraft rule-based NER surfaces exists when there is a need to change the domain and/or language of data.

1.1 Indian Language NER

Besides the above-mentioned works, there are other existing works. The majority of the languages covered include English, most of the European languages, and some of the Asian languages such as Chinese, Japanese, and Korean. India is a multilingual country with great linguistic and cultural diversities. People speak in 22 different official languages that are derived from almost all the dominant linguistic families. However, the works related to NER in Indian languages have started to emerge only very recently. Named Entity (NE) identification in Indian languages is more difficult and challenging compared to others due to a number of facts.

— Lack of capitalization information, which is an important cue for NE identification in English.
— Indian names are more diverse in nature and many of these appear in the dictionary as common nouns.
— Free word order nature of the Indian languages.
— Resource-constrained environment, that is, non-availability of corpus, annotated corpus, name dictionaries, morphological analyzers, part of speech (POS) taggers, etc., in the required measure.

Most of the works in the area of NER related to Indian languages cover a few languages such as Bengali, Hindi, and Telugu. An unsupervised approach for NER in Bengali is described in Ekbal and Bandyopadhyay [2007], where they reported two different NER models, one using lexical contextual patterns and the other using linguistic features along with the same set of lexical contextual patterns learned from an unlabeled corpus. A HMM-based supervised NER system was reported in Ekbal et al. [2007], where more contextual information was introduced during emission probabilities and NE suffixes were used for handling the unknown words. More recent works in the area of Bengali NER are based on CRF [Ekbal et al. 2008], SVM [Ekbal and Bandyopadhyay 2008a], and voting [Ekbal and Bandyopadhyay 2009].

A CRF-based Hindi NER system can be found in Li and McCallum [2004] that uses feature induction to automatically construct the features that most increase the conditional likelihood. Other works for Hindi NER can be found in Patel et al. [2009] using rules and in Saha et al. [2008] using a hybrid feature set (linguistic and statistical) based ME approach. Srikanth and Murthy [2008] developed an NER system for Telugu and tested it on several datasets from the Eenaadu and Andhra Prabha newspaper corpora for person, location, and organization tags. Gali et al. [Shishtla et al. 2008] developed a CRF-based system for English, Telugu, and Hindi. They suggested that a character n-gram based approach is more effective than word-based models to increase the recall of NER systems. Some more systems on NER in Indian languages, especially in Bengali, Hindi, Telugu, Oriya, and Urdu using different approaches are reported in the IJCNLP-08 Shared Task on NER for South and South East Asian Languages (NERSSEAL)[1]. Other than these languages, a CRF-based Tamil NER system was proposed in Vijayakrishna and Sobha [2008] for the tourism domain. This approach can take care of morphological inflections of NEs and can handle nested tagging with a hierarchical tagset containing 106 tags.

1.2 Motivation and Background of the Present Work

Classifier ensemble[2] is a new direction of machine learning. In the literature [Florian et al. 2003; Ekbal and Bandyopadhyay 2009], it has been shown that the combination of multiple classifiers could be more effective compared to any individual one for NER.

The main idea behind classifier ensemble is that ensembles are often much more accurate than the individual classifiers that make them up. Usually the members of an ensemble are generated either by applying a single learning algorithm (i.e., using homogeneous classifiers) [Dietterich 2002] or using different learning algorithms (i.e.,

[1]See http://ltrc.iiit.ac.in/ner-ssea-08.
[2]We use "ensemble classifier" and "classifier ensemble" interchangeably.

using heterogeneous classifiers) over a dataset [Wolpert 1992]. Two basic approaches to combine the outputs of several classifiers are majority voting and weighted voting [Dietterich 2002]. The existing classifier ensemble techniques can be grouped into subsampling the training examples such as bagging [Breiman 1996] and boosting [Freund and Schapire 1995a], manipulating the input features [Cherkauer 1996], manipulating the output target such as Error Correcting Output Codes (ECOC) [Dietterich and Bakiri 1995], and injecting randomness in the learning algorithm [Kolen and Pollack 1991].

A major factor in classifier ensemble is that the individual classifiers should be as diverse as possible. In the well-known ensemble techniques such as bagging and boosting, such diversities are achieved by manipulating the training examples in order to generate multiple hypotheses. The base classifier is trained several times, each time with a different subset of the training examples and thus creating different classifiers. Finally there should be a method to combine the outputs of these sets of classifiers. This method is effective for unstable learning algorithms such as neural networks, decision trees, etc. In the case of bagging, at each run the learning algorithm is trained with a training set that consists of a sample of m training examples drawn randomly with replacement from the original training set of m items. The basic idea behind boosting [Freund and Schapire 1995a] is to train a series of diverse classifiers and to iteratively focus on the hard-to-learn training examples. It is again divided into two steps. In the first step a number of classifiers are generated with various versions of the training samples, and then in the second step the classifiers are combined together to produce a more powerful one.

AdaBoost [Freund and Schapire 1995b], short for Adaptive Boosting, is a very popular machine learning algorithm. This is a meta-algorithm, and can be used in conjunction with many other learning algorithms to improve their performance. AdaBoost is also adaptive in the sense that subsequent classifiers built are tweaked in favor of those instances misclassified by previous classifiers. The main drawback associated with AdaBoost is that it is sensitive to noisy data and outliers; however, in some problems it can be less susceptible to the overfitting problem than most learning algorithms.

AdaBoost calls a weak classifier repeatedly in a series of rounds $t = 1, \ldots, T$. For each call a distribution of weights D_t is updated that indicates the importance of examples in the dataset for the classification. On each round, the weight of each incorrectly classified example is increased (or alternatively, the weights of all correctly classified examples are decreased), so that the new classifier focuses more on those examples. In this way the T individual classifiers are formed. These classifiers are then combined together into a single system using some weighted voted approach, where the weight of vote depends on the error rate of the individual classifier.

Stacking [Wolpert 1992] is another important classifier ensemble technique which basically follows a layered architecture. At the very first level, classifiers are trained using the original dataset and each classifier outputs a prediction for each token. Successive layers receive the predictions of the layer immediately preceding it as an input. Finally at the top level, a single classifier, also called meta-classifier, outputs the final prediction. Again the overall performance depends on the individual classifiers as well as on the effective classifier selection at the second level. Another problem of stacking is how to obtain the right combination of base-level classifiers and the meta-classifier especially in relation to each specific dataset. Error-correcting-code (ECOC) is another general technique for constructing a good ensemble of classifiers. Here each class is encoded as an L-bit code vector. When the L classifiers are applied to classify a new sample, their predictions are combined into a L-bit string. But it is again very tricky to choose the appropriate codewords L for each class.

Thus, it can be noted that all the existing ensemble techniques should have a way of combining the decisions of a set of classifiers. Existing approaches combine the outputs of all classifiers either by using majority voting or by using weighted voting. The weights of votes depend on the error rate/performance of individual classifiers. But none of the existing techniques quantifies the amount of vote for each output class in each classifier. However, in reality, in an ensemble system all the classifiers are not good at detecting all types of output classes. For example, in case of NER, some classifiers could be most effective at detecting person names whereas some are good to detect location names. In case of weighted voting, weights of voting should vary among the different output classes in each classifier. The weight should be high for that particular output class for which the classifier performs well. Otherwise, the weight should be low for the output class for which its output is not very reliable. So, it is a crucial issue to select the appropriate weights of votes for all the classes per classifier. In this article, we develop a genetic algorithm (GA) based method to find the appropriate weights of votes for each output class in each classifier. The method is explained below with a more general example:

Suppose there are four classifiers C_1, C_2, C_3, and C_4; and three classes cl_1, cl_2, and cl_3. Let us assume that depending on the training set and the set of features used, classifier C_1 is more efficient to classify the points in class cl_1. Similarly, depending on the configuration, classifiers C_2, C_3, and C_4 are more efficient in classifying the points in classes cl_2, cl_3, and cl_1, respectively. Let the weights of votes of classifier C_1 for three classes are $\alpha_{1,1}$, $\alpha_{1,2}$, and $\alpha_{1,3}$, respectively; $\alpha_{i,j}$ denotes the weight of vote of classifier i for output class j. In the case of Adaboost/bagging/boosting, $\alpha_{1,1} = \alpha_{1,2} = \alpha_{1,3}$. But this is not desirable in almost all real-world problems. In most of the practical applications, the reliabilities of predictions vary among the various classes in any classifier. By the proposed GA-based approach we can determine the appropriate values of αs such that $\alpha_{1,1} > \alpha_{1,2}, \alpha_{1,3}$. Similarly for C_2, C_3, and C_4, we get $\alpha_{2,2} > \alpha_{2,1}, \alpha_{2,3}$; $\alpha_{3,3} > \alpha_{3,1}, \alpha_{3,2}$ and $\alpha_{4,1} > \alpha_{4,2}, \alpha_{4,3}$. Please note that $\alpha_{1,1}$ may or may not be equal to $\alpha_{4,1}$. Depending on the effectiveness of the classifiers C_1 and C_4, the proposed GA-based approach will determine these values automatically. Another important note is that GA-based approach can automatically find $\alpha_{1,2}, \alpha_{1,3}, \alpha_{2,1}, \alpha_{2,3}, \alpha_{3,1}, \alpha_{3,2}, \alpha_{4,2}, \alpha_{4,3} \geq 0$. This implies that though C_1 is more confident to classify class cl_1, it is also eligible to vote for the other classes. The key advantages (or, novelty) of our proposed approach over the existing ensemble techniques are twofold, that is, (i) unlike previous ensembles (such as stacking), our GA-based approach does not require any technique to select the most suitable subset of classifiers from a set of base classifiers. In contrast, our proposed technique does not discard any classifier during ensemble and (ii) rather than assigning the same weights to all the classes in a classifier, like any other existing techniques, it effectively finds the proper weights of all the eligible classes depending upon the prediction confidence.

The underlying motivations behind our work are as follows.

(1) We wanted to develop a new method of combining the outputs of many classifiers. The base classifiers can be generated using the existing approaches like bagging or boosting or AdaBoost. Several different versions of any classification technique such as ME, CRF, or SVM can be developed using bagging or boosting. Thereafter the decisions can be combined together to form an ensemble using our proposed approach.
(2) Existing approaches don't quantify the amount of votes for all the classes in each classifier. However, in practical situations, the reliability of predictions varies among the different output classes for a particular classifier. In this article we propose a GA-based approach which is able to determine the appropriate

amount of vote for each output class in each classifier. This way the approach is truly novel.

(3) Another important motivation of the proposed approach is the use of GA for searching the appropriate weighted vote based classifier ensemble. Existing approaches such as AdaBoost are sensitive to noisy data and outliers. When the number of base classifiers is high, it is difficult to find the best combination of classifiers. No exhaustive search technique can be used because search space is too high in these cases. Here the search capability of GAs are used to find the global optimal solution as it is quite effective in rapid global search of large, non-linear, and poorly understood spaces. Thus the proposed method is less error-prone and always finds the best combination of classifiers.

The effectiveness of classifier ensemble has been explored in the NLP applications such as noun-phrase segmentation and NER. Carrears et al. [2002] reported an ensemble system in the CoNLL-2002 shared task that achieved the the recall, precision, and *F*-measure values of 81.4%, 81.38%, and 81.39%, respectively, for Spanish and 76.29%, 77.83%, and 77.05%, respectively, for Dutch. They used ECOC for combining different binary classifiers, based on AdaBoost. Wu et al. [2003] presented stacking and voting methods for combining strong classifiers such as boosting, SVM, and TBL in CoNLL-2003 shared task [Tjong Kim Sang and De Meulder 2003]. They obtained the recall, precision, and *F*-measure of 82.31%, 83.06%, and 82.69%, respectively for the English dataset. Florian et al. [2003] reported a NER system by combining four heterogenous classifiers, namely HMM, ME, transformation-based learning, and robust linear classifier. This system showed the best performance in CoNLL-2003 shared task [Tjong Kim Sang and De Meulder 2003] with the help of unannotated data and an additional NE tagger. In Indian languages, Ekbal and Bandyopadhyay [2009] presented a voted NER system for Bengali by combining three different classifiers such as ME, CRF and SVM. They used word-level and contextual features, gazetteers, post-processing techniques, and unlabeled data to improve the performance in each of the classifiers as well as of the overall system. On the other hand, our system (i) is based on a relatively simpler set of features, easily derivable for many languages, and (ii) does not use any additional domain dependent resources and/or tools, but still achieves the state-of-the-art performance.

A very preliminary version of the present approach is available at Ekbal and Saha [2010], where only ME was used as a base classifier. But in the current article we use heterogenous classifiers such as ME, CRF, and SVM as the underlying classification techniques, elaborately explain the algorithm with more details, evaluate it for four different Indian languages and English, and compare the performance with the three conventional baseline models as well as other existing techniques. We have also shown the effects of ensemble size and training set size on the overall system performance. In addition, we also develop a single objective optimization based feature selection technique. The classifiers trained with the feature sets identified by the feature selection technique achieve higher performance than the corresponding best individual classifier. This feature selection technique is used as another baseline to compare with the proposed classifier ensemble.

1.3 Overview of the Present Work

In this article, we propose a novel technique for classifier ensemble based on GA and evaluate it for NER. We use ME, CRF, and SVM frameworks as the base classifiers. Depending on the different combinations of the available features and/or feature templates, several versions of these classifiers are generated. All the classifiers make use of a set of features that are language independent in nature, and can be easily obtained for almost all the languages with a little effort. Thereafter, GA is used to search for the appropriate weighted vote based classifier ensemble. Instead of searching for the best performing individual classifiers, our main focus is to investigate the appropriate weights for voting. We use real encoding. Weights for voting of each classifier for every output classes are encoded in a chromosome. Mutation and crossover operators are modified accordingly. A new mutation operator is used to handle real encoding. Adaptive mutation and crossover operators are used to accelerate the convergence of GA. We also use elitism.

The proposed approach is evaluated for four resource-poor languages, namely Bengali, Hindi, Telugu, and Oriya. In terms of native speakers, Bengali is the fifth popular language in the world, second in India and the national language in Bangladesh. Hindi is the third most spoken language in the world and the national language of India. Telugu and Oriya are the other two popular languages and predominantly spoken in the southern and eastern parts of India, respectively. We manually annotate a portion, containing approximately 250,000 wordforms, of the Bengali news corpus [Ekbal and Bandyopadhyay 2008c] with a coarse-grained NE tagset of four tags namely, PER (Person name), LOC (Location name), ORG (Organization name), and MISC (Miscellaneous name). We also use the IJCNLP-08 NER on South and South East Asian Languages (NERSSEAL)[3]. Shared Task data of around 100K wordforms that were originally annotated with a fine-grained tagset of twelve tags. For Hindi, Telugu, and Oriya, we use the datasets of the NERSSEAL shared task. Evaluation results yield the recall, precision and *F*-measure values of 92.08%, 92.22%, and 92.15%, respectively for Bengali; 96.07%, 88.63%, and 92.20%, respectively for Hindi; 78.82%, 91.26%, and 84.59%, respectively for Telugu; and 88.56%, 89.98%, and 89.26%, respectively for Oriya. Finally, we evaluate the proposed approach for the CoNLL-2003 shared task [Tjong Kim Sang and De Meulder 2003] English data. Evaluation yields the overall recall, precision, and *F*-measure values of 88.72%, 88.64%, and 88.68%, respectively. This is comparable to that of the best system [Florian et al. 2003] of CoNLL-2003 shared task. The best system showed the recall, precision, and *F*-measure values of 88.54%, 88.99%, and 88.76%, respectively, with a classifier-combination experimental framework that used four diverse classifiers. It should be noted that they made use of more complex set of features, gazetteers, and an additional NE tagger. But they did not provide any information about how their system performs without any domain knowledge and/or external resources. In contrast, our proposed algorithm is based on a very simple set of features and does not use any domain knowledge but shows reasonably good performance. Results for all the languages show that the vote-based classifier ensemble identified by GA outperforms all the individual classifiers as well as the three conventional baseline ensembles, out of which one is based on traditional majority voting and the other two are based on weighted voting. Comparisons of the proposed method to some existing ensemble techniques such as stacking [Wolpert 1992] and ECOC (Error correcting output code) [Dietterich and Bakiri 1995] are also shown. For Bengali, we have also compared our proposed technique with a QBC (Query by committee) [Seung et al. 1992] based active learning method.

In a part of the article, we also formulate the problem of feature selection in any classifier under the single objective optimization framework. Thereafter, a GA-based technique is used to solve this problem. This is used as another baseline. Experimental results also suggest that our proposed method performs superior to this baseline.

[3]See http://ltrc.iiit.ac.in/ner-ssea-08.

The main contributions and/or advantages of our work are listed below.

— A new method of classifier ensemble, founded on the principle of single objective optimization technique, is proposed. The proposed technique is based on GA that quantifies the amount of voting weights for each class in each classifier. We tried to establish that such classifier ensemble is capable to increase the classification quality by a large margin compared to the conventional ensemble methods.
— To the best of our knowledge, use of GA for classifier ensemble is a novel contribution, particularly in the area of NLP and especially for NER.
— The proposed technique can be replicated for any resource-poor language very easily due to its language independent nature. In the present work, it is evaluated for four resource-poor languages such as Bengali, Hindi, Telugu, and Oriya.
— The proposed framework is a very general approach. In the present work, we use the proposed approach to solve the problem of NER. But the proposed approach can be effectively used to solve the other types of well-known classification problems such as Part-of-Speech (PoS)-tagging, question-answering, etc.
— Note, that our work proposes a novel way of combining the available classifiers. Thus, the performance of the existing works (e.g., Ekbal and Bandyopadhyay [2009], Florian et al. [2003] etc.) can be further improved with our proposed framework.
— In a part of the article we also formulate the problem of feature selection under the single objective optimization framework, and propose a solution to it using GA. This baseline showed inferior performance compared to the proposed classifier ensemble.

The rest of the paper is organized as follows. The base classifiers are briefly described in Section 2. Section 3 depicts the set of NE features that we use for NER in four leading Indian languages, namely Bengali, Hindi, Telugu, and Oriya as well as for English. The problem of weighted vote-based classifier ensemble is formulated in Section 4. We discuss our proposed GA-based ensemble technique elaborately in Section 5. Section 6 reports the detailed evaluation results along with necessary discussions for these languages. The single objective optimization based feature selection technique is described in Section 7. Finally, we conclude with the future work roadmaps in Section 8.

2. BASE CLASSIFIERS FOR NER

We use Maximum Entropy (ME), Conditional Random Field (CRF), and Support Vector Machine (SVM) as the base classifiers to construct the ensemble system based on weighted voting. Brief descriptions of these classifiers are presented below.

2.1 Maximum Entropy Framework for NER

The ME framework estimates probabilities based on the principle of making as few assumptions as possible, other than the constraints imposed. Such constraints are derived from the training data, expressing some relationships between features and outcome. The probability distribution that satisfies the above property is the one with the highest entropy. It is unique, agrees with the maximum likelihood distribution, and has the exponential form

$$P(t|h) = \frac{1}{Z(h)} exp\left(\sum_{j=1}^{n} \lambda_j f_j(h,t)\right) \tag{1}$$

where t is the NE tag, h is the context (or history), $f_j(h,t)$ are the features with associated weight λ_j, and $Z(h)$ is a normalization function.

The problem of NER can be formally stated as follows. Given a sequence of words $w_1, \ldots, w_n$, we want to find the corresponding sequence of NE tags $t_1, \ldots, t_n$, drawn from a set of tags T, which satisfies

$$P(t_1, \ldots, t_n | w_1, \ldots, w_n) = \prod_{i=1,2\ldots,n} P(t_i | h_i) \tag{2}$$

where h_i is the context for the word w_i.

The features are, in general, binary valued functions, which associate a NE tag with various elements of the context. For example,

$$\begin{aligned} f_j(h,t) &= 1 \text{ if word}(h) = \text{sachIn and } t = \text{Person name} \\ &= 0 \text{ otherwise} \end{aligned}$$

We use the OpenNLP Java based MaxEnt package [4] for the computation of the values of the parameters λ_j. This allows us to concentrate on selecting the features which best characterize the problem instead of worrying about assigning the relative weights to the features. We use the Generalized Iterative Scaling [Darroch and Ratcliff 1972] algorithm to estimate the MaxEnt parameters.

2.2 Conditional Random Field Framework for NER

Conditional Random Fields (CRFs) [Lafferty et al. 2001] are undirected graphical models, a special case of which corresponds to conditionally trained probabilistic finite state automata. Being conditionally trained, these CRFs can easily incorporate a large number of arbitrary, non-independent features while still having efficient procedures for non-greedy finite-state inference and training.

CRF is used to calculate the conditional probability of values on designated output nodes given values on other designated input nodes. The conditional probability of a state sequence $s =< s_1, s_2, \ldots, s_T >$ given an observation sequence $o =< o_1, o_2, \ldots, o_T >$ is calculated as

$$P_{\wedge}(s|o) = \frac{1}{Z_o} \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right) \tag{3}$$

where $f_k(s_{t-1}, s_t, o, t)$ is a feature function whose weight, λ_k, is to be learned via training. The values of the feature functions may range between $-\infty, \ldots + \infty$, but typically they are binary. To make all conditional probabilities sum up to 1, we must calculate the normalization factor

$$Z_o = \sum_s \exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K} \lambda_k \times f_k(s_{t-1}, s_t, o, t)\right) \tag{4}$$

which as in HMMs, can be obtained efficiently by dynamic programming.

To train a CRF, the objective function to be maximized is the penalized log-likelihood of the state sequences given the observation sequences:

$$L_{\wedge} = \sum_{i=1}^{N} \log\left(P_{\wedge}(s^{(i)} | o^{(i)})\right) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \tag{5}$$

[4]See http://maxent.sourceforge.net/.

where $\{< o^{(i)}, s^{(i)} >\}$ is the labeled training data. The second sum corresponds to a zero-mean, σ^2 -variance Gaussian prior over parameters, which facilitates optimization by making the likelihood surface strictly convex. Here, we set parameters λ to maximize the penalized log-likelihood using Limited-memory BFGS [Sha and Pereira 2003], a quasi-Newton method that is significantly more efficient and which results in only minor changes in accuracy due to changes in λ.

When applying CRFs to the NER problem, an observation sequence is a token of a sentence or document of text and the state sequence is its corresponding label sequence.

A feature function $f_k(s_{t-1}, s_t, o, t)$ has a value of 0 for most cases and is only set to be 1 when s_{t-1}, s_t are certain states and the observation has certain properties. We have used the C^{++} based CRF^{++} package [5], a simple, customizable, and open source implementation of CRF for segmenting or labeling sequential data.

2.3 Support Vector Machine Framework for NER

In the field of NLP, Support Vector Machines (SVMs) [Vapnik 1995] are applied to text categorization and are reported to have achieved high accuracy without falling into over-fitting even though with a large number of words taken as the features [Joachims 1999; Taira and Haruno 1999]. Suppose, we have a set of training data for a two-class problem: $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, where $\mathbf{x}_i \in R^D$ is a feature vector of the i-th sample in the training data and $y \in \{+1, -1\}$ is the class to which $\mathbf{x}_i$ belongs. In their basic form, a SVM learns a linear hyperplane that separates the set of positive examples from the set of negative examples with maximal margin (the margin is defined as the distance of the hyperplane to the nearest of the positive and negative examples). In basic SVMs framework, we try to separate the positive and negative examples by the hyperplane written as

$$(\mathbf{w}.\mathbf{x}) + b = 0 \quad \mathbf{w} \in \mathbf{R}^n, b \in \mathbf{R}.$$

SVMs find the "optimal" hyperplane (optimal parameter $\overline{w}, b$) which separates the training data into two classes precisely.

The linear separator is defined by two elements: a weight vector $\mathbf{w}$ (with one component for each feature), and a bias b which stands for the distance of the hyperplane to the origin. The classification rule of a SVM is

$$sgn(f(\mathbf{x}, \mathbf{w}, b)) \tag{6}$$

$$f(\mathbf{x}, \mathbf{w}, b) = < \mathbf{w}.\mathbf{x} > +b \tag{7}$$

being $\mathbf{x}$ the example to be classified. In the linearly separable case, learning the maximal margin hyperplane $(\mathbf{w}, b)$ can be stated as a convex quadratic optimization problem with a unique solution: minimize $||\mathbf{w}||$, subject to the constraints (one for each training example):

$$y_i(< \mathbf{w}.x_i > +b) \geq 1 \tag{8}$$

The SVM model has an equivalent dual formulation, characterized by a weight vector α and a bias b. In this case, α contains one weight for each training vector,

[5]See http://crfpp.sourceforge.net.

indicating the importance of this vector in the solution. Vectors with non null weights are called support vectors. The dual classification rule is

$$f(\mathbf{x}, \alpha, b) = \sum_{i=1}^{N} y_i \alpha_i < \mathbf{x}_i . \mathbf{x} > +b \tag{9}$$

The α vector can be calculated also as a quadratic optimization problem. Given the optimal α^* vector of the dual quadratic optimization problem, the weight vector $\mathbf{w}^*$ that realizes the maximal margin hyperplane is calculated as

$$\mathbf{w}^* = \sum_{i=1}^{N} y_i \alpha_i^* \mathbf{x}_i \tag{10}$$

The b^* has also a simple expression in terms of $\mathbf{w}^*$ and the training examples $(\mathbf{x}_i, y_i)_{i=1}^{N}$.

The advantage of the dual formulation is that efficient learning of non-linear SVM separators, by introducing *kernel functions*. Technically, a kernel function calculates a dot product between two vectors that have been (non linearly) mapped into a high dimensional feature space. Since there is no need to perform this mapping explicitly, the training is still feasible although the dimension of the real feature space can be very high or even infinite.

By simply substituting every dot product of $\mathbf{x}_i$ and $\mathbf{x}_j$ in dual form with any kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$, SVMs can handle non-linear hypotheses. Among the many kinds of kernel functions available, we will focus on the d-th polynomial kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i . \mathbf{x}_j + 1)^d$$

Use of d-th polynomial kernel function allows us to build an optimal separating hyperplane which takes into account all combination of features up to d.

Support Vector Machines have advantage over conventional statistical learning algorithms from the following two aspects:

(1) SVMs have high generalization performance independent of dimension of feature vectors.
(2) SVMs can carry out their learning with all combinations of given features without increasing computational complexity by introducing the kernel function.

We develop our system using SVM [Joachims 1999; Vapnik 1995] which perform classification by constructing an N-dimensional hyperplane that optimally separates data into two categories. We have used YamCha[6] toolkit, an SVM based tool for detecting classes in documents and formulating the NER task as a sequential labeling problem. Here, the *pairwise multi-class decision* method and the *polynomial kernel function* are used. We use the TinySVM-0.07[7] classifier.

3. NAMED ENTITY FEATURES

The main features for the NER task are identified based on the different possible combinations of available word and tag contexts. We use the following features for constructing the various classifiers based on the ME, CRF, and SVM frameworks. Most

[6]See http://chasen-org/ taku/software/yamcha/.
[7]See http://cl.aist-nara.ac.jp/ taku-ku/software/TinySVM.

of these features are language independent in nature and can be easily obtained for almost all the languages. The features are language independent in the sense that these don't need any domain dependent resources and/or language specific rules for their generation. We also define a semantically motivated feature that proved to be very effective to improve the overall system performance.

(1) Context words: These are the preceding and succeeding words of the current word. This is based on the observation that surrounding words carry effective information for the identification of NEs.
(2) Word suffix and prefix: Fixed length (say, n) word suffixes and prefixes are very effective for identifying NEs and work well for the highly inflective Indian languages. Actually, these are the fixed length character strings stripped either from the rightmost or from the leftmost positions of the words. For example, the suffixes of length up to three characters of the word "*ObAmA*" [Obama] are "*A*", "*mA*", and "*AmA*". Henceforth, all the Bengali glosses are written in ITRANS notation [8]. The prefixes of length up to three characters of the word "*ObAmA*" [Obama] are "*O*", "*Ob*", and "*ObA*". If the length of the corresponding word is less than or equal to $n-1$ then the feature values are not defined and denoted by ND. The feature value is also not defined (ND) if the token itself is a punctuation symbol or contains any special symbol or digit. This feature is included with the observation that NEs share some common suffixes and/or prefixes.
(3) First word: This is a binary valued feature that checks whether the current token is the first word of the sentence or not. We consider this feature with the observation that the first word of the sentence is most likely a NE. We observed that NEs generally appear in the first position of the sentence in news-wire data.
(4) Length of the word: This binary valued feature checks whether the number of characters in a token is less than a predetermined threshold value (here, set to 5). This feature is defined with the observation that very short words are most probably not the NEs.
(5) Infrequent word: This is a binary valued feature that checks whether the current word appears in the training set very frequently or not. We compile a list of most frequently occurring words from the training set by defining an appropriate threshold value. This threshold value does vary depending upon the size of the training set. In the present work, we set the threshold values to 10, 15, 7, and 10 for Bengali, Hindi, Telugu, and English, respectively. A binary valued feature "INFRQ" fires if and only if the word does not appear in this list. We include this feature as the frequently occurring words are most likely not the NEs.
(6) Last word of sentence: This feature checks whether the word is the last word of a sentence or not. In Indian languages, verbs generally appear in the last position of the sentence. Indian languages follow subject-object-verb structure. This feature distinguishes NEs from the verbs.
(7) Capitalization: This is a binary valued feature that determines whether the word starts with a capital letter or not. It is found to be an useful feature for English. In the present work, this feature is used only for English as Indian languages do not have any capitalization cues.
(8) Part-of-Speech (POS) information: POS information of the current and/or the surrounding tokens(s) are effective for NE identification. We use a SVM-based POS tagger [Ekbal and Bandyopadhyay 2008b] for Bengali, Hindi, and Telugu. This

[8]See Bengali glosses are written using ITRANS notation (http://www.aczone.com/itrans/).

POS tagger was developed with a tagset[9] of 27 POS tags, defined as part of the Indian languages. The POS tagger has been trained with the Bengali, Hindi, and Telugu data, obtained through participations in the NLPAI_Contest06[10] and SPSAL2007[11]competitions. In this particular work, we evaluate the SVM-based POS tagger with a coarse-grained tagset that contains only three tags: namely Nominal, PREP (Postpositions), and Other. Postpositions are considered as they often appear after the NEs. Due to the non-availability of POS-tagged datasets in Oriya, we couldn't use this feature. For English, the POS information was already provided with the datasets of CoNLL-2003 shared task.

(9) Chunk information: This is useful for NE identification. Here, this is used only for English due to the non-availability of any chunker in Indian languages. The possible values of this feature are B-NP/I-NP (if the token is at the beginning/internal position of a noun phrase), B-VP/I-VP (if the token is at the beginning/internal position of a verb phrase), etc.

(10) Digit features: Several digit features are defined depending upon the presence and/or the number of digits and/or symbols in a token. These features are digitComma (token contains digit and comma), digitPercentage (token contains digit and percentage), digitPeriod (token contains digit and period), digitSlash (token contains digit and slash), digitHyphen (token contains digit and hyphen) and digitFour (token consists of four digits only).

(11) Semantic feature: This feature is semantically motivated. We consider all unigrams in contexts $w_{i-3}^{i+3} = w_{i-3} \ldots w_{i+3}$ of w_i (crossing sentence boundaries) for the entire training data. We convert tokens to lowercase and remove stopwords, numbers, and punctuation symbols. We define a feature vector of length 20 using the 20 most frequent content words. Given a classification instance, the feature corresponding to token t is set to 1 iff the context w_{i-3}^{i+3} of w_i contains t.

(12) Dynamic NE information: This is the output label(s) of the previous token(s). The value of this feature is determined dynamically at run time. This feature is used for ME and SVM models. For CRF, we consider the bi-gram template that calculates all the feature combinations of the current and previous tokens.

4. PROBLEM FORMULATION

Suppose, the N number of available classifiers is denoted by $C_1, \ldots, C_N$. Let, $\mathcal{A} = \{C_i : i = 1; N\}$. Suppose, there are M number of output classes. The weighted vote-based classifier ensemble selection problem is then stated as follows:

Find the weights of votes V per classifier which will optimize some functions $F_i(V); i \leq 1$. Here, V is a real array of size $N \times M$. $V(i, j)$ denotes the weight of vote of the i^{th} classifier for the j^{th} class. More weight is assigned for that particular class for which the classifier is more confident; whereas the output classes for which the classifier is less confident are given less weight. $V(i, j) \in [0, 1]$ denotes the degree of confidence of the i^{th} classifier for the j^{th} class. These weights are used while combining the outputs of classifiers using weighted voting. Here, F_is are some classification quality measures of the combined weighted vote-based classifier. The particular type of problem like NER has mainly three different kinds of classification quality measures, namely recall, precision and F-measure. Thus, $F \in$ {recall, precision, F-measure}. Here we have chosen $F = F$-measure.

[9]See http://shiva.iiit.ac.in/SPSAL2007/iiit_tagset_guidelines.pdf.
[10]See http://ltrc.iiitnet/nlpaicontest06/.
[11]See http://shiva.iiit.ac.in/SPSAL2007/.

```
Begin
  1. $t = 0$
  2. initialize population $P(t)$ /* $Popsize = |P|$ */
  3. for $i = 1$ to $Popsize$
       compute fitness $P(t)$
  4. $t = t + 1$
  5. if termination criterion achieved go to step 10
  6. select $(P)$
  7. crossover $(P)$
  8. mutate $(P)$
  9. go to step 3
  10. output best chromosome and stop
End
```

Fig. 1. Basic steps of GA.

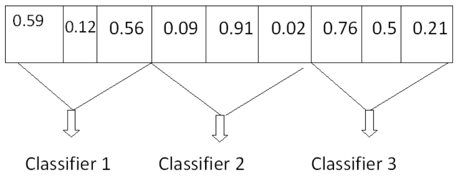

Fig. 2. Chromosome representation.

5. PROPOSED APPROACH

The proposed GA-based classifier ensemble method is described below. The basic steps of the proposed approach closely follow those of the conventional GA as shown in Figure 1.

5.1 Overview of Genetic Algorithm

Genetic Algorithms (GAs) [Goldberg 1989] are randomized search and optimization techniques guided by the principles of evolution and natural genetics, having a large amount of implicit parallelism. GAs perform search in complex, large, and multi-modal landscapes, and provide near-optimal solutions for objective or fitness function of an optimization problem. In GAs, the parameters of the search space are encoded in the form of strings called *chromosomes*. A collection of such strings is called a *population*. Initially, a random population is created, which represents different points in the search space. An *objective* or a *fitness* function is associated with each string that represents the degree of *goodness* of the string. Based on the principle of survival of the fittest, a few of the strings are selected and each is assigned a number of copies that go into the mating pool. Biologically inspired operators, such as *crossover*, and *mutation*, are applied on these strings to yield a new generation of strings. The process of selection, crossover, and mutation continues for a fixed number of generations or until a termination condition is satisfied.

5.2 String Representation and Population Initialization

Suppose there are M number of available classifiers and O number of output classes. Then the length of the chromosome is $M \times O$. Each chromosome encodes the weights of votes for possible O output classes for each classifier. Please note that chromosome represents the available classifiers along with their weights for each class. It bears the same meaning for all types of models such as ME, CRF, and SVM. As an example, the encoding of a particular chromosome is represented in Figure 2, where $M = 3$

and $O = 3$ (i.e., a total 9 votes is possible). The chromosome represents the following ensemble:
The weights of votes for three different output classes in classifier 1 are 0.59, 0.12, and 0.56, respectively. Similarly, weights of votes for 3 different output classes are 0.09, 0.91, and 0.02, respectively, in classifier 2 and 0.76, 0.5 and 0.21, respectively, in classifier 3.

In the figure, classifiers 1, 2, and 3 could be either of the same type (i.e., ME, CRF, or SVM) or different types (i.e., one/more may be of one category such as CRF and the rest are of different categories) of classification methods. We use real encoding that randomly initializes the entries of each chromosome by a real value (r) between 0 and 1. Here, $r = \frac{rand()}{RAND\_MAX+1}$. If the population size is P then all the P number of chromosomes of this population are initialized in the above way.

5.3 Fitness Computation

Initially, the F-measure values of all the available classifiers are calculated using three-fold cross validation on the available training data. Then, we execute the following steps to compute the fitness value of each chromosome.

(1) Suppose there are total M number of classifiers. Let, the overall F-measure values of these M classifiers be F_i, $i = 1 \dots M$.
(2) Initially, the training data is divided into three parts. Each classifier is trained using 2/3 of the training data and tested with the remaining 1/3 part. We have M possible output classes (each from a different classifier) for each word in the 1/3 training data. Now for the ensemble classifier, the label of the output class for each word in the 1/3 training data is determined using the weighted voting of these M classifiers' outputs. The weight of the output label provided by the i^{th} classifier is equal to $I(m, i)$. Here, $I(m, i)$ is the entry of the chromosome corresponding to m^{th} classifier and i^{th} class. The combined score of a particular class for a particular word w is:

$$f(c_i) = \sum I(m, i) \times F_m, \quad \forall m = 1 \text{ to } M \text{ and } op(w, m) = c_i \tag{11}$$

Here, $op(w, m)$ denotes the output label provided by the m^{th} classifier for the word w. The class receiving the maximum combined score is selected as the joint decision.
(3) The overall F-measure value of this ensemble is calculated for the 1/3 training data.
(4) Steps 2 and 3 are repeated 3 times to perform three-fold cross validation.
(5) The average F-measure value of this three-fold cross validation is used as the fitness value of the particular chromosome. The objective is to maximize this fitness value using the search capability of GA.

5.4 Selection

During each successive generation, a proportion of the existing population is selected to generate a new generation. Individual solutions are selected through a fitness-based process where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions.

In this article, we use roulette wheel selection. Here, the fitness function associated with each chromosome is used to associate a probability of selection with each individual chromosome. If f_i is the fitness of individual i in the population, its probability of being selected is

$$p_i = \frac{f_i}{\Sigma_{j=1}^{N} f_j},$$

where N is the number of individuals in the population.

This selection process has resemblance to a roulette wheel in a casino. Usually, a proportion of the wheel is assigned to each of the possible selections based on their fitness value. This could be achieved by dividing the fitness of a selection by the total fitness of all the selections, thereby normalizing them to 1. Then a random selection is made similar to how the roulette wheel is rotated. Thus, in case of roulette wheel selection, chromosomes with higher fitness values are less likely to be eliminated but there is still a chance that they may be.

5.5 Crossover

Here, we use the normal single point crossover [Holland 1975]. As an example, let the two chromosomes be
$P1$: 0.24 0.16 0.54 0.87 0.66 0.76 0.01 0.88 0.21
$P2$: 0.12 0.09 0.89 0.71 0.65 0.82 0.69 0.43 0.15

At first a crossover point has to be selected randomly between 1 to 9 (length of the chromosome) by generating some random number between 1 and 9. Let the crossover point, here, be 4. Then after crossover, the two new offsprings are
$O1$: 0.24 0.16 0.54 0.87 0.65 0.82 0.69 0.43 0.15 (taking the first four positions from $P1$ and rest from $P2$)
$O2$: 0.12 0.09 0.89 0.71 0.66 0.76 0.01 0.88 0.21 (taking the first four positions from $P2$ and rest from $P1$)

Crossover probability is selected adaptively as in Srinivas and Patnaik [1994]. Crossover probabilities are computed as follows. Let f_{max} be the maximum fitness value of the current population, $\overline{f}$ be the average fitness value of the population and $f^{'}$ be the larger of the fitness values of the solutions to be crossed. Then the probability of crossover, μ_c, is calculated as
$\mu_c = k_1 \times \frac{(f_{max} - f^{'})}{(f_{max} - \overline{f})}$, if $f^{'} > \overline{f}$,
$\mu_c = k_3$, if $f^{'} \leq \overline{f}$.

Here, as in Srinivas and Patnaik [1994], the values of k_1 and k_3 are kept equal to 1.0. Note that when f_{max}=$\overline{f}$, then $f^{'} = f_{max}$ and μ_c will be equal to k_3. The aim behind this adaptation is to achieve a trade-off between exploration and exploitation in a different manner. The value of μ_c is increased when the better of the two chromosomes to be crossed is itself quite poor. In contrast when it is a good solution, μ_c is low so as to reduce the likelihood of disrupting a good solution by crossover.

5.6 Mutation

Each chromosome undergoes mutation with a probability μ_m. The mutation probability is also selected adaptively for each chromosome as in Srinivas and Patnaik [1994].

The expression for mutation probability, μ_m, is given below:
$\mu_m = k_2 \times \frac{(f_{max} - f)}{(f_{max} - \overline{f})}$ if $f > \overline{f}$,
$\mu_m = k_4$ if $f \leq \overline{f}$.

Here, values of k_2 and k_4 are kept equal to 0.5. This adaptive mutation helps GA to come out of local optimum. When GA converges to a local optimum, that is, when

$f_{max} - \overline{f}$ decreases, μ_c and μ_m both will be increased. As a result, GA will come out of local optimum. It will also happen for the global optimum and may result in disruption of the near-optimal solutions. This may distract GA to converge to the global optimum. The μ_c and μ_m will get lower and higher values for high and low fitness solutions, respectively. While the high fitness solutions aid in the convergence of the GA, the *low* fitness solutions prevent the GA from getting stuck at a local optimum. The use of elitism will also keep the best solution intact. For a solution with the maximum fitness value, μ_c and μ_m are both zero. The best solution in a population is transferred undisrupted into the next generation. Together with the selection mechanism, this may lead to an exponential growth of the solution in the population and may cause premature convergence. To overcome the above stated problem, a default mutation rate (of 0.02) is kept for every solution in the proposed approach.

Here, each position in a chromosome is mutated with probability μ_m in the following way. The value is replaced with a random variable drawn from a Laplacian distribution, $p(\epsilon) \propto e^{-\frac{|\epsilon-\mu|}{\delta}}$, where the scaling factor δ sets the magnitude of perturbation. Here, μ is the value at the position which is to be perturbed. The scaling factor δ is chosen equal to 0.1. The old value at the position is replaced with the newly generated value. By generating a random variable using Laplacian distribution, there is a non-zero probability of generating any valid position from any other valid position while probability of generating a value near the old value is more.

5.7 Termination Condition

In this approach, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the above classifier ensemble problem. Elitism is implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus on termination, this location contains the best classifier ensemble.

6. DATASETS, EXPERIMENTAL SETUP AND RESULTS

In this section, we present the detailed discussions on the datasets, experimental setup, and evaluation results for all the languages along with necessary discussions.

6.1 Datasets for NER

Indian languages are resource-constrained in nature. For NER, we use a Bengali news corpus [Ekbal and Bandyopadhyay 2008c] developed from the archive of a leading Bengali newspaper available on the Web. We manually annotate a portion containing approximately 250K wordforms with a coarse-grained NE tagset of four tags namely, PER (Person name), LOC (Location name), ORG (Organization name) and MISC (Miscellaneous name). The Miscellaneous name includes date, time, number, percentages, monetary expressions, and measurement expressions. The data is collected mostly from the national, states, sports domains and the various subdomains of districts of the particular newspaper. This annotation was carried out by one of the authors and verified by an expert. We also use the IJCNLP-08 NER on South and South East Asian Languages (NERSSEAL)[12] Shared Task data of around 100,000 wordforms that were originally annotated with a fine-grained tagset of twelve tags. This data is mostly from the agriculture and scientific domains. For Hindi, Telugu and Oriya, we use the datasets obtained from the NERSSEAL shared task. The underlying reason to adopt the finer NE tagset in the shared task was to use the NER system in various

[12]See http://ltrc.iiit.ac.in/ner-ssea-08.

Table I. NE Tagset for Indian Languages (IJCNLP-08 NERSSEAL Shared Task Tagset)

NE Tag	Meaning	Example
NEP	Person name	*sachIna*/NEP, *sachIna ramesha tenDUlkara* / NEP
NEL	Location name	*kolkAtA*/NEL, *mahatmA gAndhi roDa* / NEL
NEO	Organization name	*yadabpUra bishVbidyAlYa*/NEO, *bhAbA eytOmika risArcha sentAra* / NEO
NED	Designation	*cheYArmAn*/NED, *sA.msada*/NED
NEA	Abbreviation	*bi e*/NEA, *ci em di a*/NEA, *bi je pi*/NEA, *Ai.bi.em*/ NEA
NEB	Brand	*fYAntA*/NEB
NETP	Title-person	*shrImAna*/NED, *shrI*/NED, *shrImati*/NED
NETO	Title-object	*AmericAn biUti*/NETO
NEN	Number	*10*/NEN, *dasha*/NEN
NEM	Measure	*tina dina*/NEM, *p.NAch keji*/NEM
NETE	Terms	*hidena markbha madela*/NETE, *kemikYAla riYYAkchYAna*/NETE
NETI	Time	*10 i mAgha 1402* / NETI, *10 ema*/NETI

Table II. Tagset Mapping Table

IJCNLP-08 shared task tag	Coarse-grained tag	Meaning
NEP	PER	Person name
NEL	LOC	Location name
NEO	ORG	Organization name
NEN, NEM, NETI	MISC	Miscellaneous name
NED, NEA, NEB, NETP, NETE	O	Other than NEs

NLP applications, particularly in machine translation. The IJCNLP-08 NERSSEAL shared task tagset is shown in Table I. One important aspect of the shared task was to identify and classify the maximal NEs as well as the nested NEs, that is, the constituent parts of a larger NE. But, the training data were provided with the type of the maximal NE only. For example, *mahatmA gAndhi roDa*(Mahatma Gandhi Road) was annotated as location and assigned the tag "NEL" even if *mahatmA* (Mahatma) and *gAndhi* (Gandhi) are NE title person (NETP) and person name (NEP), respectively. The task was to identify *mahatmA gAndhi roDa* as a NE and classify it as NEL. In addition, *mahatmA* and *gAndhi* had to be recognized as NEs of categories NETP (Title person) and NEP (Person name), respectively.

In the present work, we consider only the tags that denote person names (NEP), location names (NEL), organization names (NEO), number expressions (NEN), time expressions (NETI), and measurement expressions (NEM). The NEN, NETI, and NEM tags are mapped to the MISC tag that denotes miscellaneous entities. Other tags of the shared task are mapped to the "other-than-NE" category denoted by "O". Hence, the tagset mapping now becomes as shown in Table II.

In order to properly denote the boundaries of NEs, four basic NE tags are further divided into the format I-TYPE (TYPE→PER/LOC/ORG/MISC) which means that the word is inside a NE of type TYPE. Only if two NEs of the same type immediately follow each other, the first word of the second NE will have tag B-TYPE to show that it starts a new NE. For example, the name *mahatmA gAndhi*[Mahatma

Table III. Statistics of the Datasets

Language	#words in training	# NEs in training	# words in test	# NEs in test
Bengali	312,947	37,009	37,053	4,413
Hindi	444,231	43,021	58,682	3,005
Telugu	57,179	4,470	6,847	662
Oriya	93,573	4,477	2,183	206

Gandhi] is tagged as *mahatmA*[Mahatma]/I-PER *gAndhi*[Gandhi]/I-PER. But, the names *mahatmA gAndhi*[Mahatma Gandhi] *rabIndrAnAth thAkur* [Rabindranath Tagore] are to be tagged as: *mahatmA*[Mahatma]/I-PER *gAndhi*[Gandhi]/I-PER *rabIndrAnAth*[Rabindranath]/B-PER *thAkur*[Tagore]/I-PER, if they appear sequentially in the text. This is the standard IOB format that was followed in the CoNLL-2003 shared task [Tjong Kim Sang and De Meulder 2003]. In order to report the evaluation results, we randomly select a portion of each dataset as the gold standard test data. Some statistics of training and test datasets for the Indian languages are presented in Table III. For English NER, we use the CoNLL-2003 shared task data.

6.2 Experimental Setup

We set the following parameter values for GA: population size = 100, number of generations = 50, probability of mutation and crossover are determined adaptively.

We define three different baseline classifier ensemble techniques as follows.

(1) *Baseline 1*. In this baseline model, all the individual classifiers are combined together into a final system based on the majority voting of the output class labels. If all the outputs differ then anyone is selected randomly.
(2) *Baseline 2*. All the individual classifiers are combined with the help of a weighted voting approach. In each classifier, weight is calculated based on the average *F*-measure value of the three-fold cross validation on the training data. The final output label is selected based on the highest weighted vote.
(3) *Baseline 3*. For each classifier, the average *F*-measure value of each output class is computed from the three-fold cross validation on the training data. The weight of any classifier is set to the average *F*-measure value of the corresponding class, assigned by it to any word token.

For the purpose of comparison three state-of-the-art ensemble techniques, namely stacking [Wolpert 1992], QBC (Query by committee)[Seung et al. 1992] and ECOC (Error correcting Output Codes) [Dietterich and Bakiri 1995] are executed on the obtained classifiers. In stacking, SVM is used as a meta-classifier for all the languages, namely Bengali, Hindi, Telugu, Oriya, and English.

A QBC-based active learning technique is used for Bengali. We use a portion of the unlabeled Bengali news corpus [Ekbal and Bandyopadhyay 2008c] for active learning. This portion is consisting of 35,000 news documents that contain approximately 10 million wordforms. Here, we form a committee by selecting the two top-performing classifiers from each of ME, CRF, and SVM models (see Section 6.3). The decision obtained from this committee is used to improve the performance of the best performing classifier. The available 35,000 documents are divided into 35 equal subsets. Initially, all the available classifiers are trained using the available training data and evaluated with the first subset of unlabeled data. Most uncertain samples of this test data are selected for manual verification by the user. The user is provided with the surrounding context of previous three and next three words to determine the correct labels of the uncertain samples. After verification, these samples are added to the initial training data. We considered the samples to be uncertain if the output labels of all the classifiers do not agree to a single decision. This process is repeated 35 times. Finally, the best individual classifier is retrained using the resultant training data and evaluated with the available test data. We could not employ this technique for other Indian languages due to the non-availability of unlabeled data. For ECOC, initially we generate binary classifiers for each NE class. Binary classifiers are generated using ME, CRF, and SVM for the Indian languages and for English. Thereafter, ECOC [Dietterich and Bakiri 1995] method is applied to solve the multi-class problem. The code matrix is generated exhaustively and minimum Hamming distance-based method is applied to determine the appropriate class label of each test instance.

6.3 Results and Discussions

We build a number of different ME, CRF, and SVM models by considering the various combinations of the available NE features and/or feature templates. In this particular work, we construct various classifiers by considering the various subsets of the following set of features: various context window within the previous three and next three words, that is, $w_{i-3}^{i+3} = w_{i-3} \ldots w_{i+3}$ of w_i, word suffixes and prefixes of length upto three (3+3 different features) or four (4+4 different features) characters, POS information of the current and/or surrounding token(s), first word, length, infrequent word, last word in the sentence, several digit features, semantic feature, and dynamic NE information.

A feature vector consisting of the features as described above is extracted for each word in the NE-tagged corpus. Now, we have a training data in the form (W_i, T_i), where, W_i is the i^{th} word and its feature vector and T_i is its corresponding output class. For CRF, we consider various combinations from the set of feature templates as given by $F_1 = \{w_{i-m}, \ldots, w_{i-1}, w_i, w_{i+1}, \ldots, w_{i+n}$; combination of w_{i-1} and w_i; combination of w_i and w_{i+1}; feature vector consisting of all other features of w_i; and B (bigram feature template)$\}$.

Please note that in CRF, the bigram feature template represents the combination of current and preceding output labels. For Bengali, we generate 152 different models using ME classifier by varying the available features. Some 21 of these classifiers are shown in Table IV. Varying the available features and/or feature templates, we construct many CRF and SVM based classifiers, out of which nine CRF-based and eight SVM-based classifiers are shown in Table IV.[13] The CRF-based model exhibits best performance with the overall recall, precision, and F-measure values of 89.42%, 90.55%, and 89.98%, respectively. Thereafter, we apply our proposed GA-based approach to determine the appropriate classifier ensemble. Overall evaluation results of this ensemble techniques along with the best individual classifier, three different baseline ensembles and some other existing ensemble techniques are reported in Table V. Results show that the overall performance attained by the GA-based classifier ensemble determined by the proposed algorithm performs better than the best performing individual classifier with the increments of 2.66, 1.67, and 2.17 percentage recall, precision, and F-measure points, respectively. The proposed GA-based ensemble technique performs reasonably better than three *baseline* models. It demonstrates the overall performance improvement with the increments of 6.79, 6.05, and 5.00 percentage F-measure points over Baseline 1, Baseline 2, and Baseline 3, respectively.

[13]Note, that we use the same set of features for Bengali, Hindi, and Telugu, but we report the results of only the best performing classifiers, and so these feature combinations could vary for each language in the respective evaluation tables.

Table IV. Evaluation Results of the Individual Classifiers for Bengali. Here, the Following Abbreviations Are Used: "CW": Context Words, "PS": Size of the Prefix, "SS": Size of the Suffix, "POS": POS Information, "WL": Word Length, "IW": Infrequent Word, "PW": Position of the Word, "FW": First Word, "DI": "Digit-Information", "NE": Dynamic NE Information, "FT": Feature Template, "Sem": Semantic Feature, $-i, j$: Words Spanning from the i^{th} Left to the j^{th} Right Position, "$-i$": Previous *i* Number of Tokens, Current Token is at 0^{th} Position, "P": Previous Token, "C": Current Token, "N": Next Token, "B": Bigram Feature Template, "r": Recall, "p": Precision, "F": *F*-Measure, X: Denotes the Presence of the Corresponding Feature for the Current Token (We Report Percentages)

Classifier	Context	FW	PS	SS	POS	WL	IW	PW	DI	NE/FT	Sem	r	p	F
ME_1	-2,2	X	3		X				X	-1	X	83.69	87.92	85.75
ME_2	-2,1	X	3		X				X	-1	X	83.92	87.87	85.85
ME_3	-1,1	X	3		X				X	-1	X	84.05	86.96	85.48
ME_4	-1,2	X	3		X				X	-1	X	84.03	86.19	85.09
ME_5	-2,2	X	3	3	X				X	-1	X	84.87	88.09	86.45
ME_6	-2,1	X	3	3	X				X	-1	X	85.82	87.28	86.54
ME_7	-2,0	X	3	3	X				X	-1	X	83.92	87.36	85.60
ME_8	-1,1	X	3	3	X				X	-1	X	84.48	86.63	85.54
ME_9	-1,2	X	3	3	X				X	-1	X	85.12	87.52	86.30
ME_{10}	0,2	X	3	3	X				X	-1	X	84.76	86.69	85.71
ME_{11}	-3,3	X	3	3	X				X	-1	X	84.12	88.10	86.07
ME_{12}	-2,2	X	4	3	X				X	-1	X	83.44	88.23	85.77
ME_{13}	-2,1	X	4	3	X				X	-1	X	83.62	88.15	85.83
ME_{14}	-1,1	X	4	3	X				X	-1	X	83.71	87.09	85.37
ME_{15}	-1,2	X	4	3	X				X	-1	X	83.71	87.84	85.73
ME_{16}	-2,2	X	3	4	X				X	-1	X	83.80	87.89	85.80
ME_{17}	-2,1	X	3	4	X				X	-1	X	84.21	87.87	86.00
ME_{18}	-2,0	X	3	4	X				X	-1	X	83.46	87.05	85.22
ME_{19}	-1,1	X	3	4	X				X	-1	X	83.78	86.56	85.15
ME_{20}	-1,2	X	3	4	X				X	-1	X	84.17	87.52	85.81
ME_{21}	-3,3	X	3	4	X				X	-1	X	83.19	87.74	85.41
CRF_1	-2,2	X	4	4	X	X	X	X	X	B	X	88.67	89.91	89.29
CRF_2	-3,3	X	4	4	X	X	X	X	X	B	X	88.49	89.71	89.09
CRF_3	-3,2	X	4	4	X	X	X	X	X	B	X	88.51	89.79	89.15
CRF_4	-1,1	X	4	4	X	X	X	X	X	B	X	88.49	89.26	88.87
CRF_5	-2,2	X	4	4	X	X	X	X	X	B		74.85	83.03	78.73
CRF_6	-2,2	X	3 (P & C)	3 (P & C)	X	X	X	X	X	B	X	89.42	90.55	89.98
CRF_7	-2,2	X	3	3	X	X	X	X	X	B	X	89.06	90.10	89.57
CRF_8	-2,2	X	3 (C & N)	3 (C & N)	X	X	X	X	X	B	X	88.46	89.75	89.10
CRF_9	-1,1	X	3	3	X	X	X	X	X	B	X	87.85	89.25	88.55
SVM_1	$-1, 1$	X	4	4	X	X	X	X	X	-1	X	87.58	87.31	87.44
SVM_2	$-2, 2$	X	4	4	X	X	X	X	X	-1	X	87.49	87.59	87.54
SVM_3	$-2, 2$	X	4	4	X	X	X	X	X	-2	X	87.13	87.19	87.16
SVM_4	$-2, 2$	X	3	3	X	X	X	X	X	-2	X	87.29	87.39	87.34
SVM_5	$-2, 2$	X	2	2	X	X	X	X	X	-2	X	86.52	86.95	86.73
SVM_6	$-2, 2$	X	2	2	X	X	X	X		-2	X	86.63	86.95	86.79
SVM_7	$-2, 2$	X	4	4	X	X	X	X	X	-2	X	75.93	83.32	79.45
SVM_8	$-2, 2$	X	3	3	X					-2		65.65	76.71	70.75

The relatively lower performance in the baseline ensembles is due to the blind combination of all the available classifiers. This fact also points in support of our underlying assumption that quantification of voting for each class in each classifier is very important.

Statistical analysis of variance (ANOVA) [Anderson and Scolve 1978] is performed in order to examine whether the proposed ensemble technique really outperforms the best individual classifier, three baseline ensembles and three other existing ensemble techniques. Our proposed technique is based on GA, a heuristic based search and optimization technique. The final results provided by GAs largely depend on the seed value of the random variables and values of parameters. Here, we have executed the proposed approach 10 times and the best among these is reported in Table VI. But, the three baselines always provide the same results in every run. This is due to the fact

Table V. Overall Results for Bengali (We Report in Percentages)

Classification Scheme	recall	precision	*F*-measure
Best individual classifier	89.42	90.55	89.98
Baseline 1	84.83	85.90	85.36
Baseline 2	85.25	86.97	86.10
Baseline 3	86.97	87.34	87.15
Stacking	90.17	91.74	90.95
ECOC	89.78	90.89	90.33
QBC	90.01	91.09	90.55
GA-based ensemble	92.08	92.22	92.15

Table VI. Estimated Marginal Means and Pairwise Comparison Between the Proposed GA-Based Approach, Individual Classifier, and Several Other Ensembles

Evaluation criterion	Tech nique(I)	Comp.	Mean Diff. (I-J)	Significance value
F-measure	GA-based approach	Individual Classifier	2.17 ± 0.013	$1.1623e-009$
F-measure	GA-based approach	*Baseline 1*	6.79 ± 0.014	$3.3990e-009$
F-measure	GA-based approach	*Baseline 2*	6.05 ± 0.011	$8.6386e-010$
F-measure	GA-based approach	*Baseline 3*	5.00 ± 0.009	$5.5376e-010$
F-measure	GA-based approach	Stacking	1.20 ± 0.009	$5.5376e-010$
F-measure	GA-based approach	ECOC	1.82 ± 0.004	$3.2176e-010$
F-measure	GA-based approach	QBC	1.60 ± 0.054	$6.6176e-010$

that all the available classifiers are ensembled in the baselines. So for ANOVA analysis, we consider ten different runs (in maximum of the cases results are almost same) of GA, whereas the same results are used ten times for three baselines. Thereafter, ANOVA analysis is carried out on these outputs. ANOVA tests show that the differences in mean recall, precision, and *F*-measure values are statistically significant as p value is less than 0.05 in each of the cases. Evaluation results of the ANOVA analyses are shown in Table VI. Results reveal that the GA based approach truly performs better than the best individual classifiers, three baseline approaches, and three other ensemble techniques.

Thereafter, the proposed approach is evaluated on Hindi and Telugu datasets. The various classifier combinations are reported in Table VII and Table X for Hindi and Telugu, respectively. We use the same set of features as Bengali. The overall performance of the best individual classifier, three baseline ensembles, single objective GA-based ensemble, and the two other existing ensemble techniques are presented in Table VIII and Table XI for Hindi, and Telugu, respectively. For Hindi, CRF model exhibits the best performance whereas SVM model shows the highest performance for Telugu. The proposed GA-based ensemble yields the overall recall, precision and *F*-measure values as 96.07%, 88.63%, and 92.20%, respectively for Hindi, and 78.82%, 91.26%, and 84.59%, respectively for Telugu. Results show that the overall performance of the classifier ensemble determined by the proposed algorithm is better than all the other models. For Hindi, it shows the improvement of *F*-measure values by 2.80%, 17.51%, 8.56%, 7.61%, 2.00%, and 1.57%, respectively, over the best individual classifier, three *baseline* approaches and two existing ensemble techniques.

Results for Telugu show that the overall performance attained by the GA-based ensemble is reasonably better (an improvement of 1.40%, 13.27%, and 6.89% recall, precision and *F*-measure values, respectively) than the best individual classifier. We

Table VII. Evaluation Results of the Individual Classifiers for Hindi. Here, the Abbreviations Are the Same as Bengali (We Report Percentages)

Classifier	Context	FW	PS	SS	POS	WL	IW	PW	DI	NE/FT	Sem	r	p	F
ME_1	-2,2	X	3		X				X	-1	X	81.73	89.09	85.25
ME_2	-2,1	X	3		X				X	-1	X	82.13	88.71	85.29
ME_3	-1,1	X	3		X				X	-1	X	82.99	89.02	85.90
ME_4	-1,2	X	3		X				X	-1	X	82.33	89.52	85.78
ME_5	0,2	X	3		X				X	-1	X	82.59	89.34	85.84
ME_6	0,1	X	3		x				X	-1	X	85.19	87.67	86.41
ME_7	-2,2	X	3	3	X				X	-1	X	84.16	92.21	86.00
ME_8	-2,1	X	3	3	X				X	-1	X	83.03	90.00	86.37
ME_9	-2,0	X	3	3	X				X	-1	X	82.03	86.79	85.28
ME_{10}	-1,1	X	3	3	X				X	-1	X	82.63	89.66	86.00
ME_{11}	-1,2	X	3	3	X				X	-1	X	82.53	90.03	86.12
ME_{12}	0,2	X	3	3	X				X	-1	X	82.43	89.73	85.93
ME_{13}	0,1	X	3	3	X				X	-1	X	83.72	89.52	86.53
ME_{14}	-3,3	X	3	3	X				X	-1	X	81.23	90.91	85.80
ME_{15}	-2,2	X	3	4	X				X	-1	X	81.99	90.35	85.97
ME_{16}	-2,1	X	3	4	X				X	-1	X	82.69	89.73	86.07
ME_{17}	-2,0	X	3	4	X				X	-1	X	82.03	88.91	85.33
ME_{18}	-1,1	X	3	4	X				X	-1	X	83.03	89.65	86.21
ME_{19}	-1,2	X	3	4	X				X	-1	X	82.63	90.27	86.28
ME_{20}	0,2	X	3	4	X				X	-1	X	82.36	89.56	85.81
ME_{21}	0,1	X	3	4	X				X	-1	X	83.29	89.43	86.26
ME_{22}	-3,3	X	3	4	X				X	-1	X	81.19	90.74	85.71
CRF_1	-2,2	X	4	4	X	X	X	X	X	B	X	88.02	89.36	88.68
CRF_2	-3,3	X	4	4	X	X	X	X	X	B	X	87.42	88.90	88.15
CRF_3	-3,2	X	4	4	X	X	X	X	X	B	X	87.92	89.35	88.63
CRF_4	-1,1	X	4	4	X	X	X	X	X	B	X	88.62	89.78	89.20
CRF_5	-2,2	X	4	4	X	X	X	X	X	B		66.82	80.32	72.95
CRF_6	-2,2	X	3 (P & C)	3 (P & C)	X	X	X	X	X	B	X	87.72	89.17	88.44
CRF_7	-2,2	X	3	3	X	X	X	X	X	B	X	88.72	90.10	89.40
CRF_8	-2,2	X	3 (C & N)	3 (C & N)	X	X	X	X	X	B	X	87.59	89.07	88.32
CRF_9	-1,1	X	3	3	X	X	X	X	X	B	X	87.45	89.08	88.26
SVM_1	$-1, 1$	X	4	4	X	X	X	X	X	-1	X	87.95	88.33	88.14
SVM_2	$-2, 2$	X	4	4	X	X	X	X	X	-1	X	88.38	89.24	88.81
SVM_3	$-2, 2$	X	4	4	X	X	X	X	X	-2	X	85.55	86.27	85.91
SVM_4	$-2, 2$	X	3	3	X	X	X	X	X	-2	X	84.89	85.63	85.26
SVM_5	$-2, 2$	X	2	2	X	X	X	X	X	-2	X	83.72	84.65	84.18
SVM_6	$-2, 2$	X	2	2	X	X	X	X		-2	X	83.22	84.09	83.65
SVM_7	$-2, 2$	X	3	3	X	X	X	X	X	-1	X	83.79	84.46	84.12
SVM_8	$-2, 2$	X	3	3	X					-2		64.65	75.71	69.74

Table VIII. Overall Results for Hindi (We Report Percentages)

Classification Scheme	recall	precision	*F*-measure
Best individual classifier	88.72	90.10	89.40
Baseline 1	63.32	90.99	74.69
Baseline 2	74.67	94.73	83.64
Baseline 3	75.52	96.13	84.59
Stacking	89.80	90.61	90.20
ECOC	90.16	91.11	90.63
GA-based ensemble	96.07	88.63	92.20

also observe the increments of 3.36, 3.76, and 3.25 percentage *F*-measure points over Baseline 1, Baseline 2 and Baseline 3, respectively. The proposed algorithm also performs superior with more than 3.83 and 3.21 percentage *F*-measure values compared to two existing ensemble techniques. Note that unlike Bengali, the baseline models based on majority voting do not perform well in comparison to weighted voting for Hindi and Telugu both. This may be due to the nature of the datasets.

Finally, the proposed system is evaluated for Oriya, one of the popularly spoken languages in the eastern part of India. The various classifier combinations are reported

Table IX. Estimated Marginal Means and Pairwise Comparison Between the Proposed GA-Based Approach, Individual Classifier and Several Other Ensembles

Evaluation criterion	Technique(I)	Comp.	Mean Diff. (I-J)	Significance value
F-measure	GA-based approach	Individual Classifier	2.80 ± 0.013	$2.9623e-009$
F-measure	GA-based approach	*Baseline 1*	17.51 ± 0.024	$2.3990e-009$
F-measure	GA-based approach	*Baseline 2*	8.56 ± 0.041	$9.6386e-010$
F-measure	GA-based approach	*Baseline 3*	7.61 ± 0.059	$6.8376e-010$
F-measure	GA-based approach	Stacking	2.00 ± 0.029	$3.2176e-010$
F-measure	GA-based approach	ECOC	1.82 ± 0.004	$3.2176e-010$

Table X. Evaluation Results of the Individual Classifiers for Telugu. Here, the Abbreviations Are the Same as Bengali (We Report Percentages)

Classifier	Context	FW	PS	SS	POS	WL	IW	PW	DI	NE/FT	Sem	r	p	F
ME_1	-2,2	X	3	-	X				X	-1	X	65.40	78.94	71.53
ME_2	-2,1	X	3	-	X				X	-1	X	65.40	78.52	71.36
ME_3	-1,1	X	3	-	X				X	-1	X	67.01	78.52	72.31
ME_4	-1,2	X	3	-	X				X	-1	X	65.40	78.11	71.19
ME_5	0,2	X	3	-	X				X	-1	X	65.54	78.84	71.58
ME_6	0,1	X	3	-	X				X	-1	X	67.16	77.50	71.96
ME_7	-2,2	X	3	3	X				X	-1	X	67.01	82.79	74.07
ME_8	-2,1	X	3	3	X				X	-1	X	67.01	82.79	74.07
ME_9	-2,0	X	3	3	X				X	-1	X	67.45	82.73	74.31
ME_{10}	-1,1	X	3	3	X				X	-1	X	69.35	82.98	75.56
ME_{11}	-1,2	X	3	3	X				X	-1	X	68.48	83.24	75.14
ME_{12}	0,2	X	3	3	X				X	-1	X	68.33	83.07	74.98
ME_{13}	0,1	X	3	3	X				X	-1	X	70.97	83.30	76.64
ME_{14}	-3,3	X	3	3	X				X	-1	X	64.52	81.94	72.19
ME_{15}	-2,2	X	4	3	X				X	-1	X	67.01	81.61	73.59
ME_{16}	-2,1	X	4	3	X				X	-1	X	68.33	82.04	74.56
ME_{17}	-2,0	X	4	3	X				X	-1	X	67.30	81.10	73.56
ME_{18}	-1,1	X	4	3	X				X	-1	X	69.79	81.37	75.14
ME_{19}	-1,2	X	4	3	X				X	-1	X	68.92	81.46	74.67
ME_{20}	0,2	X	4	3	X				X	-1	X	68.92	82.02	74.90
ME_{21}	0,1	X	4	3	X				X	-1	X	70.53	81.94	75.81
ME_{22}	-3,3	X	4	3	X				X	-1	X	65.10	81.02	72.19
ME_{23}	-1,1	X	3	4	X				X	-1	X	67.74	80.07	73.39
ME_{24}	-1,2	X	3	4	X				X	-1	X	66.72	79.96	72.74
ME_{25}	0,2	X	3	4	X				X	-1	X	66.13	81.12	72.86
ME_{26}	0,1	X	3	4	X				X	-1	X	68.48	80.10	73.84
ME_{27}	-1,1	X	4	4	X				X	-1	X	67.45	77.97	72.33
ME_{28}	0,2	X	4	4	X				X	-1	X	66.28	78.75	71.98
ME_{29}	0,1	X	4	4	X				X	-1	X	68.33	78.58	73.10
CRF_1	-2,2	X	4	4	X	X	X	X	X	B	X	74.63	75.18	74.91
CRF_2	-3,3	X	4	4	X	X	X	X	X	B	X	73.75	74.30	74.02
CRF_3	-3,2	X	4	4	X	X	X	X	X	B	X	74.34	74.89	74.61
CRF_4	-1,1	X	4	4	X	X	X	X	X	B	X	76.39	76.96	76.67
CRF_5	-2,2	X	4	4	X	X	X	X	X	B		37.98	94.19	54.13
CRF_6	-2,2	X	3 (P & C)	3 (P & C)	X	X	X	X	X	B	X	73.31	75.08	74.18
CRF_7	-2,2	X	3	3	X	X	X	X	X	B	X	75.66	77.36	76.50
CRF_8	-2,2	X	3 (C & N)	3 (C & N)	X	X	X	X	X	B	X	73.61	75.04	74.32
CRF_9	-1,1	X	3	3	X	X	X	X	X	B	X	73.17	74.70	73.93
SVM_1	$-1, 1$	X	4	4	X	X	X	X	X	-1	X	77.42	77.99	77.70
SVM_2	$-2, 2$	X	4	4	X	X	X	X	X	-1	X	77.42	77.99	77.70
SVM_3	$-2, 2$	X	4	4	X	X	X	X	X	-2	X	74.34	74.89	74.61
SVM_4	$-2, 2$	X	3	3	X	X	X	X	X	-2	X	73.46	74.00	73.73
SVM_5	$-2, 2$	X	2	2	X	X	X	X	X	-2	X	71.99	72.53	72.26
SVM_6	$-2, 2$	X	2	2	X	X	X	X		-2	X	73.61	74.15	73.88
SVM_7	$-2, 2$	X	4	4	X	X	X	X	X	-2	X	75.55	76.85	76.19
SVM_8	$-2, 2$	X	3	3	X					-2	-	73.90	74.45	74.17

Table XI. Overall Results for Telugu (We Report Percentages)

Classification Scheme	recall	precision	F-measure
Best individual classifier	77.42	77.99	77.70
Baseline 1	60.12	87.39	71.23
Baseline 2	71.87	92.33	80.83
Baseline 3	72.22	93.10	81.34
Stacking	77.65	84.12	80.76
ECOC	77.96	85.12	81.38
GA-based ensemble	78.82	91.26	84.59

Table XII. Estimated Marginal Means and Pairwise Comparison Between the Proposed GA-Based Approach, Individual Classifier, and Several Other Ensembles for Telugu

Evaluation criterion	Technique(I)	Comp.	Mean Diff. (I-J)	Significance value
F-measure	GA-based approach	Individual Classifier	6.89 ± 0.023	$3.7623e - 009$
F-measure	GA based approach	*Baseline 1*	3.36 ± 0.089	$6.3990e - 009$
F-measure	GA-based approach	*Baseline 2*	3.76 ± 0.022	$7.1186e - 010$
F-measure	GA-based approach	*Baseline 3*	3.25 ± 0.039	$5.4576e - 010$
F-measure	GA-based approach	Stacking	3.83 ± 0.0219	$4.8176e - 010$
F-measure	GA-based approach	ECOC	3.21 ± 0.024	$6.2176e - 010$

in Table XIII. We train each classifier with the same set of language independent features, as used for other languages except POS information. Detailed evaluation results are presented in Table XIV that shows the overall recall, precision, and F-measure values as 88.56%, 89.98%, and 89.26%, respectively. Results also show that the overall performance of the proposed GA-based classifier ensemble is better (an improvement of 2.01%, 1.95%, and 1.97% in recall, precision, and F-measure values, respectively) than the best individual classifier. In comparison to three *baseline* ensembles, our proposed method performs superior by the margins of 1.63, 1.46, and 0.90 percentage F-measure points, respectively. The proposed GA-based approach also performs superior to two state-of-the-art ensemble techniques, namely stacking and ECOC. Evaluation for all the languages suggest that rather than blindly combining all the available classifiers, it is more effective to find out the appropriate weights of voting for each class in each classifier.

6.4 Results on CoNLL-2003 Shared Task English DataSet

Here, we evaluate our proposed technique on the benchmark English dataset of CoNLL 2003 shared task [Tjong Kim Sang and De Meulder 2003]. Initially, the system is tuned on the development data, and then blind evaluation is performed on the test data to report the evaluation results. The CoNLL-2003 shared task English dataset had 203,621, 46,435, and 51,362 tokens for training, development, and test sets, respectively. The different classifier combinations obtained by varying the feature sets and/or feature templates in ME, CRF, and SVM (as the underlying classification techniques) are reported in Table XV. The best performance is obtained for the CRF based model that yields the overall recall, precision and F-measure values of 86.16%, 86.47%, and 86.31%, respectively. Overall evaluation results are presented in Table XVI. It shows the overall recall, precision, and F-measure values of 88.72%, 88.64%, and 88.68%, respectively, which are better than the best individual classifier, three baseline ensemble techniques and two state-of-the-art ensemble techniques.

Table XIII. Evaluation Results of Oriya for Various Feature Subsets. Here, Abbreviations Are Same as Bengali (We Report Percentages)

Classifier	Context	FW	PS	SS	WL	IW	PW	DI	NE/FT	Sem	r	p	F
ME_1	-2,2	X	3	-				X	-1	X	71.40	84.94	77.58
ME_2	-2,1	X	3	-				X	-1	X	71.40	84.52	77.41
ME_3	-1,1	X	3	-				X	-1	X	73.01	84.52	78.34
ME_4	-1,2	X	3	-				X	-1	X	71.40	84.11	77.24
ME_5	0,2	X	3	-				X	-1	X	71.54	84.84	77.62
ME_6	0,1	X	3	-				X	-1	X	73.16	83.50	77.99
ME_7	-2,2	X	3	3				X	-1	X	73.01	88.79	80.13
ME_8	-2,1	X	3	3				X	-1	X	73.01	88.79	80.13
ME_9	-2,0	X	3	3				X	-1	X	73.45	88.73	80.37
ME_{10}	-1,1	X	3	3				X	-1	X	75.35	88.98	81.60
ME_{11}	-1,2	X	3	3				X	-1	X	74.48	89.24	81.19
ME_{12}	0,2	X	3	3				X	-1	X	74.33	89.07	81.04
ME_{13}	0,1	X	3	3				X	-1	X	76.97	89.30	82.68
ME_{14}	-3,3	X	3	3				X	-1	X	70.52	87.94	78.27
ME_{15}	-2,2	X	4	3				X	-1	X	73.01	87.61	79.65
ME_{16}	-2,1	X	4	3				X	-1	X	74.33	88.04	80.61
ME_{17}	-2,0	X	4	3				X	-1	X	73.30	87.10	79.61
ME_{18}	-1,1	X	4	3				X	-1	X	75.79	87.37	81.17
ME_{19}	-1,2	X	4	3				X	-1	X	74.92	87.46	80.71
ME_{20}	0,2	X	4	3				X	-1	X	74.92	88.02	80.94
ME_{21}	0,1	X	4	3				X	-1	X	76.53	87.94	81.84
ME_{22}	-3,3	X	4	3				X	-1	X	71.10	87.02	78.26
ME_{23}	-1,1	X	3	4				X	-1	X	73.74	86.07	79.43
ME_{24}	-1,2	X	3	4				X	-1	X	72.72	85.96	78.79
ME_{25}	0,2	X	3	4				X	-1	X	72.13	87.12	78.92
ME_{26}	0,1	X	3	4				X	-1	X	74.48	86.10	79.87
ME_{27}	-1,1	X	4	4				X	-1	X	73.45	83.97	78.36
ME_{28}	0,2	X	4	4				X	-1	X	72.28	84.75	78.02
ME_{29}	0,1	X	4	4				X	-1	X	74.33	84.58	79.12
CRF_1	-2,2	X	4	4	X	X	X	X	B	X	83.19	84.62	83.90
CRF_2	-3,3	X	4	4	X	X	X	X	B	X	82.35	83.76	83.05
CRF_3	-3,2	X	4	4	X	X	X	X	B	X	84.03	85.47	84.75
CRF_4	-1,1	X	4	4	X	X	X	X	B	X	86.55	88.03	87.29
CRF_5	-2,2	X	4	4	X	X	X	X	B		82.35	84.48	83.40
CRF_6	-2,2	X	3 (P & C)	3 (P & C)	X	X	X	X	B	X	84.03	85.47	84.75
CRF_7	-2,2	X	3	3	X	X	X	X	B	X	84.03	85.47	84.75
CRF_8	-2,2	X	3 (C & N)	3 (C & N)	X	X	X	X	B	X	84.03	85.47	84.75
CRF_9	-1,1	X	3	3	X	X	X	X	B	X	86.55	88.03	87.29
SVM_1	$-1, 1$	X	4	4	X	X	X	X	-1	X	82.35	83.05	82.70
SVM_2	$-2, 2$	X	4	4	X	X	X	X	-1	X	82.35	83.05	82.70
SVM_3	$-2, 2$	X	4	4	X	X	X	X	-2	X	82.35	83.05	82.70
SVM_4	$-2, 2$	X	3	3	X	X	X	X	-2	X	82.35	82.35	82.35
SVM_5	$-2, 2$	X	2	2	X	X	X	X	-2	X	79.83	79.83	79.83
SVM_6	$-2, 2$	X	2	2	X	X	X		-2	X	79.83	79.83	79.83
SVM_7	$-2, 2$	X	4	4	X	X	X	X	-2	X	82.35	83.05	82.70
SVM_8	$-2, 2$	X	3	3					-2	-	47.90	85.07	61.29

Table XIV. Overall Results for Oriya (We Report Percentages)

Model	recall (in %)	precision (in %)	F-measure (in %)
Best individual classifier	86.55	88.03	87.29
Baseline 1	86.95	88.33	87.63
Baseline 2	87.12	88.50	87.80
Baseline 3	87.62	89.12	88.36
Stacking	87.90	89.53	88.71
ECOC	87.04	88.56	87.79
GA-based ensemble	88.56	89.98	89.26

Table XV. Evaluation Results of English for Various Feature Subsets. Here, “Chunk”: Chunk Information, Cap: Capitalization Information, and Other Abbreviations are the Same as Bengali (We Report Percentages)

Classifier	Context	FW	PS	SS	WL	IW	PW	POS	DI	Chunk	Cap	POS	Chunk	NE/FT	Sem	R	P	F
M_1	-3,3	X	4	3	-	-	-	X	X	X	X			-1	X	80.21	83.65	81.89
M_2	-2,2	X	3	4	-	-	-	X	X	X	X	-	-	-1	X	80.82	84.33	82.54
M_3	-2,1	X	3	4	-	-	-	X	X	X	X	-	-	-1	X	80.56	83.41	81.96
M_4	-1,2	X	4	4	-	-	-	X	X	X	X	-	-	-1	X	80.45	83.60	82.00
M_5	-2,2	X	4	3	X	-	-	X	X	X	X	-	-	-1	X	80.33	84.11	82.18
M_6	-2,1	X	4	3	X	-	-	X	X	X	X	-	-	-1	X	80.37	83.71	82.00
M_7	-1,2	X	4	3	X	-	-	X	X	X	X	-	-	-1	X	80.37	83.82	82.06
M_8	-2,2	X	4	4	X	-	-	X	X	X	X	-	-	-1	X	80.43	84.33	82.33
M_9	-2,2	X	3	4	X	X		X	X	X	X			-1	X	80.45	83.80	82.09
M_{10}	-2,2	X	4	4	X	X		X	X	X	X			-1	X	80.95	84.02	82.46
SVM_1	-2,2	X	4	4	X	X	X	X	X	X	X	-	-	1	X	85.03	85.66	85.34
SVM_2	-2,2	X	4	4	X	X	X	X	X	X	X	-1	-1	-2	X	83.84	84.29	84.07
SVM_3	-2,2	X	4	4	X	X	X	X	X	X	X	-2	-2	-2	X	83.67	84.14	83.90
SVM_4	-2,2	X	4	4	X	X	X	X	X	X	X	-2	-2	-	X	85.46	85.97	85.72
SVM_5	-3,3	X	4	4	X	X	X	X	X	X	X	-	-	-	X	85.21	85.94	85.57
SVM_6	-3,3	X	4	4	X	X	X	X	X	X	X	-3	-3	-	X	85.39	86.03	85.71
SVM_7	-4,4	X	4	4	X	X	X	X	X	X	X	-	-	-	X	85.39	86.12	85.76
SVM_8	-3,3	X	4	4	X	X	X	X	X	X	X	-	-	-2	X	84.05	84.68	84.37
CRF_1	-2,2	X	4	4	X	X	X	X	X	X	-	-	-	B	X	84.76	85.75	85.25
CRF_2	-3,3	X	4	4	X	X	X	X	X	X	X	-	-	B	X	84.63	85.87	85.24
CRF_3	-3,2	X	4	4	X	X	X	X	X	X	X	-	-	B	X	84.62	85.66	85.13
CRF_4	-1,1	X	4	4	X	X	X	X	X	X	-	-	-	B	X	83.49	84.54	84.01
CRF_5	-2,2	X	4	4	X	X	X	X	X	X				B	X	83.25	81.64	82.43
CRF_6	-2,2	X	4	4	X	X	X	X	X	X	-	-	-	B	X	84.90	85.83	85.36
CRF_7	-2,2	X	4 (P & C)	4 (P & C)	X	X	X	X	X	X	X	-	-	B	X	85.13	86.02	85.58
CRF_8	-2,2	X	3	3	X	X	X	X	X	X	X	-	-	B	X	84.23	84.98	84.60
CRF_9	-2,2	X	3	3	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.50	85.77	85.63
CRF_{10}	-3,2	X	3	3	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.48	85.89	85.68
CRF_{11}	-3,3	X	3	3	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.50	85.87	85.69
CRF_{12}	-1,1	X	3	3	X	X	X	X	X	X	X	-1,1	-1,1	B	X	84.01	84.28	84.15
CRF_{13}	-2,2	X	3	3	X	X	X	X	X	X	-	-1,1	-1,1	B	X	83.96	82.30	83.12
CRF_{14}	-2,2	X	4 (P & C)	4 (P & C)	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.74	86.03	85.89
CRF_{15}	-2,2	X	3 (P & C)	3 (P & C)	X	X	X	X	X	X	X	-1,1	-1,1	B	X	86.16	86.47	86.31
CRF_{16}	-2,2	X	3 (C & N)	3 (C & N)	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.03	85.24	85.14
CRF_{17}	-3,2	X	3	3	X	X	X	X	X	X	X	-1,1	-1,1	B	X	85.05	85.34	85.19

Table XVI. Overall Results for English (We Report Percentages)

Model	recall (in %)	precision (in %)	F-measure (in %)
Best individual classifier	86.16	85.24	86.31
Baseline 1	85.75	86.12	85.93
Baseline 2	86.20	87.02	86.61
Baseline 3	86.65	87.25	86.95
Stacking	85.93	86.45	86.18
ECOC	86.12	85.34	85.72
GA-based ensemble	88.72	88.64	88.68

Table XVII. Results by Varying the Ensemble Size for Bengali (We Report in Percentages)

Ensemble size	recall	precision	F-measure
80 (best-performing ME classifiers)	87.51	89.67	88.57
140 (best-performing ME classifiers)	87.51	89.67	88.57
152 (all ME classifiers)	87.51	89.67	88.57
161 (all ME + all CRF)	91.43	91.98	91.70
169 (ME, CRF, SVM)	92.08	92.22	92.15
180 (adding some more CRF based classifiers)	92.08	92.22	92.15

6.5 Results with Varying Ensemble Size and Training Set Size

For the purpose of illustration, we experiment with the various ensemble size for Bengali. The results are reported in Table XVII. Note that increasing the ensemble size increases the length of the chromosome and thereby increases the size of the search space. As an example, for the ensemble size = 50 and number of output classes = 8, total length of the chromosome will be 400. This, in turn, increases the size of the search space. Thus, it is very difficult to find the appropriate weight combinations for each classifier. Results also suggest that increasing the number of classifiers may not always increase the overall performance of the system. If more good performing/complimentary classifiers will be added to the total set of classifiers then the overall system performance may increase. But in general, after a certain size of the classifier ensemble, system performance does not improve with the increase of ensemble size. Table XIII shows that the system achieves the recall, precision, and F-measure values of 87.51%, 89.67%, and 88.57%, respectively, with the ensemble that is formed with the 80 best performing ME-based classifiers. But if we increase the number of classifiers to either 140 or 152, the performance does not improve. Inclusion of heterogenous classifiers improves the overall performance of the system. Though, the overall performance increases until we reach the size of the ensemble to 169. But, it is also to be noted that greater ensemble size does not deteriorate the overall performance of the system. One possible explanation behind this nature of performance is that the weights of most of the classifiers for maximum number of output classes are automatically set near zero values by the proposed GA-based method.

We also experiment by varying the size of the training set in order to observe their effect on the overall performance. Results are reported in Table XVIII with the various training sets and evaluating on the same test set. Table XVIII shows that the performance of the system is proportional with the size of the training set. But the rate of improvements are gradually decreasing per iteration.

Table XVIII. Results by Varying the Training Set Size for Bengali

Training set size	recall (in %)	precision (in %)	F-measure (in %)
100K	83.43	85.24	83.34
150K	85.74	87.79	86.75
200K	87.99	89.13	88.55
250K	89.83	90.76	90.29
300K	91.95	91.97	91.96
312K	92.08	92.22	92.15

6.6 Computational Time Requirements for Different Methods

In the experimental results we have compared our approach with three standard baseline techniques along with some existing ensembling techniques, stacking, ECOC, and QBC. We have executed the proposed technique on HP Server running red hat linux enterprize 4 operating system with 6 GB RAM and 250 GB hard disk.

For stacking, baselines and the proposed approach the first step is the same, that is, generating all the base classifiers. Here, we report the time required by the techniques for computing the next step, i.e. in case of stacking the time taken by second level classifier; in case of the proposed method the time taken by the GA based approach; in case of baselines time taken to combine the outputs. For Bengali, stacking method took 82 minutes to train and test the second level SVM classifier. Baseline approaches are the fastest as these need only the time to combine the individual classifiers. These methods took only 2, 3.2, and 4.6 minutes, respectively. Our proposed approach took 30 minutes of time to determine the final result. Again for Hindi the time taken by stacking, three baselines and the proposed approach are 74, 1.5, 2, 2.5, and 28 minutes, respectively. Thus it can be noted that while for stacking the time complexity depends on the classifier used in the second level, our approach is independent of the base classifiers used. This is an advantage of our proposed technique.

QBC involves a lot of computations, and it is the most time-consuming technique among all of our experiments. We have used this technique only for Bengali. Each ME, CRF and SVM classifiers took 72, 91, and 96 minutes, respectively, for training and evaluating with the first set of unlabeled data. The time taken for combining all the decisions was 1.7 minutes. The uncertain samples were selected in 1.2 minutes. All these steps were repeated 35 times. Two users were employed to assign the correct labels to the most uncertain samples, and they spent 2.4 hours on an average for each set of unlabeled documents. Finally, the best individual classifier took 167 minutes in total for training and testing. In contrast, in the experimental settings of our proposed method, the maximum time was spent in generating only the base classifiers. The later stages took less than 30 minutes to generate the final results.

The complexity of the proposed technique also depends on the number of individual base classifiers. With the increase of the number of base classifiers, the complexity of the proposed technique increases.

6.7 Observations

Results show that weighted vote-based classifier ensemble identified by the proposed GA-based technique performs better than the best individual classifier, three different baseline ensemble techniques, and some other state-of-the art ensemble techniques for all the datasets. This is because of the following.

(1) It is already established in machine-learning literature that proper ensemble of classifiers should always perform better in comparison to the existing base classifiers.

(2) Three baseline ensemble techniques are not able to combine the classifiers properly. As a result, in some cases, the combined model even shows inferior performance compared to the best individual classifiers.
(3) The effectiveness of GA in finding out proper weights of voting is another reason they show better performance than the existing state-of-the-art ensemble approaches.

The second observation also supports the gravity of our underlying hypothesis that proper classifier ensemble selection is, itself, a very crucial problem. Our proposed GA-based approach effectively selects the appropriate weights of voting for each class in each classifier. For Indian languages, all the results show that the proposed algorithm performs best for Bengali followed by Hindi, Oriya, and Telugu. The possible reasons could be (i). the ratios of positive (NEs) and negative examples (non-NEs) in the respective training data, that is, 1:9 (Bengali), 1:9.33 (Hindi)., 1:13 (Telugu), and 1:19 (Oriya); (ii) agglutinative nature of Telugu that may possibly require a different feature set; and (iii) presence of less number of NEs in the Oriya dataset. The proposed algorithm shows a general high precision but suffers from relatively low recall values for all these languages. This may be due to the fact that all the datasets are highly imbalanced and hence greatly biased towards the negative categories. Thus, there are not enough NE instances that could be more effective for NE identification. These need to investigate appropriate sampling strategy to make the ratios of positive and negative examples more balanced. We also investigated the effects of training set size and ensemble size on the overall performance of the system. For CoNLL-2003 shared task English dataset, the proposed technique achieves the performance comparable to the state-of-the-art systems. We have developed a single objective optimization based feature selection technique which also showed inferior performance compared to the proposed approach.

6.8 Comparisons to Other Works

It will not be fair to compare the performance of our proposed system with that of the previous proposals for Bengali [Ekbal and Bandyopadhyay 2007; Ekbal et al. 2007, 2008; Ekbal and Bandyopadhyay 2008a, 2009], Hindi [Li and McCallum 2004; Patel et al. 2009; Saha et al. 2008] and Telugu [Srikanth and Murthy 2008] as these works use (i) different experimental setup, (ii) different data sets, (iii) more complex set of features, and (iv) domain-dependent knowledge and/or resources. In contrast, our proposed algorithm is based on a relatively small set of features that can be easily obtained for almost all the languages, does not make use of any domain dependent resources, and thus can be replicated for any resource-poor language very easily. Though we use the IJCNLP-08 NERSSEAL shared task data, we convert these fine-grained annotated data to the coarse-grained forms. Thus, comparing our proposed system with that of the shared task articles [14] is also out-of-scope. Comparisons of the proposed method to some existing ensemble techniques such as stacking [Wolpert 1992] and ECOC [Dietterich and Bakiri 1995] are also shown. In addition to stacking and ECOC, we have also compared our proposed method with a QBC based [Seung et al. 1992] active learning technique for Bengali.

For the benchmark English dataset, our proposed system achieves the performance, comparable to the best performing system [Florian et al. 2003] of CoNLL-2003 shared task [Tjong Kim Sang and De Meulder 2003]. The best system [Florian et al. 2003] at CoNLL-2003 shared task demonstrated the recall, precision, and *F*-measure values of 88.54%, 88.99%, and 88.76%, respectively. They presented a classifier-combination

[14]See http://ltrc.iiit.ac.in/ner-ssea-08.

Table XIX. Comparisons with Some Existing Systems for English NER

System	F-measure (in %)
Lin and Wu [Lin and Wu 2009]	90.90
Suzuki and Isozaki [Suzuki and Isozaki 2008]	89.92
Florian et al. [Florian et al. 2003]	88.76
Our Proposed System	**88.68**
Chieu and Ng [Chieu and Ng 2003]	88.31
Klein et al. [Klein et al. 2003]	86.31
Wu et al. [Wu et al. 2003]	82.69

experimental framework for NER in which four diverse classifiers, namely robust linear classifier, ME, transformation-based learning, and HMM. They made use of a more complex set of features: gazetteer information, in the form of a list of 50,000 cities, 80,000 proper names, and 3,500 organizations. They also used the outputs of other two NE classifiers, trained on a richer tagset of 32 named categories for improving the system performance. However, they did not report any results on the test set without using any domain knowledge. In contrast, our proposed algorithm (i) makes use of a very simple set of features that can be derived for any language with a little effort, (ii) does not use any domain-dependent resources like the gazetteers etc., and (iii) does not make use of any additional NE taggers, but still achieves state-of-the-art performance, which is below 0.08 F-measure point compared to the best system of CoNLL-2003.

Until now, the best reported results for CoNLL-2003 shared task data are in Lin and Wu [Lin and Wu 2009] that proposed a semi-supervised approach for NER. In addition to word clusters, they used phrase clusters as the features and achieved significant performance improvement. They proposed an algorithm for clustering tens of millions of phrases and used the resulting clusters as features in discriminative classifiers. They collected 20 million unique queries from an anonymized query log that were found in a 700 billion token Web corpus with a minimum frequency count of 100. Thereafter, they constructed the feature vectors for 20 million phrases using the web data, executed the K-means clustering on the phrases that appeared in the CoNLL training data to obtain K centroids, and assigned each of the 20 million phrases to the nearest centroid of clusters. They used CRF as a base classifier and experimented with 48 various feature templates. They obtained the F-measure value of 90.90%, which is 1.22 points higher than our proposed system. In addition to the above mentioned two systems [Florian et al. 2003; Lin and Wu 2009], we also present the comparative results with some other well-known existing techniques in Table XIX. Suzuki and Isozaki [2008] run a baseline discriminative classifier on unlabeled data to generate pseudo examples, which are then used to train a different type of classifier for the same problem. Later on, they used the automatically labeled corpus to train HMMs. Chieu and Ng [2003] showed how the use of global information, in addition to the local ones, can improve the model performance. Our system also achieves 5.99 points higher F-measure value in comparison to the stacked, voted model proposed by Wu et al. [2003] in the CoNLL-2003 shared task.

7. FEATURE SELECTION USING SINGLE OBJECTIVE OPTIMIZATION BASED TECHNIQUE

In a machine learning approach, feature selection is an optimization problem that involves choosing an appropriate feature subset. In an ME model, appropriate feature selection is a very crucial problem and also a key issue to improve classifier's performance. However, it does not provide any method for automatic feature selection and heuristics are usually used for this task. In this section, we report a single objective

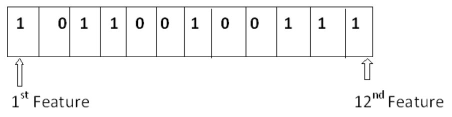

Fig. 3. Chromosome representation for GA-based feature selection.

optimization based feature selection technique based on GA [Goldberg 1989]. This is used as a baseline to compare with the performance of the proposed classifier ensemble technique. The GA-based feature selection technique is described below.

7.1 String Representation and Population Initialization

If the total number of features is F, then the length of the chromosome is F. As an example, the encoding of a particular chromosome is represented in Figure 3. Here $F = 12$ (i.e., total 12 different features are available). The chromosome represents the use of seven features for constructing a classifier (first, third, fourth, seventh, tenth, eleventh, and twelfth features). The entries of each chromosome are randomly initialized to either 0 or 1. Here, if the i^{th} position of a chromosome is 0 then it represents that i^{th} feature does not participate in constructing the classifier. Else if it is 1 then the i^{th} feature participates in constructing the classifier.

If the population size is P then all the P number of chromosomes of this population are initialized in the above way.

7.2 Fitness Computation

For the fitness computation, the following procedure is executed.

(1) Suppose there are N number of features present in a particular chromosome (i.e., there are total N number of 1s in that chromosome).
(2) Construct a classifier with only these N features.
(3) Here, initially the training data is divided into three parts. The above classifier is trained using 2/3 of the training set with the features encoded in that chromosome and tested with the remaining 1/3 part.
(4) Now, the overall F-measure value of this classifier for the 1/3 training data is calculated.
(5) Steps 3 and 4 are repeated three times to perform three-fold cross validation.
(6) The average F-measure value of this three-fold cross validation is used as the fitness value of the particular chromosome. The objective is to maximize this fitness value using the search capability of GA.

7.3 Genetic Operators

Here, roulette wheel selection is used to implement the proportional selection strategy. For crossover, normal single point crossover [Holland 1975] is used. Crossover probability is selected adaptively as in Srinivas and Patnaik [1994]. Each chromosome undergoes mutation with a probability μ_m. The mutation probability is also selected adaptively for each chromosome as in Srinivas and Patnaik [1994]. Here, mutation operator is applied to each entry of the chromosome where the entry is randomly replaced by either 0 or 1.

In this approach, the processes of fitness computation, selection, crossover, and mutation are executed for a maximum number of generations. The best string seen up to the last generation provides the solution to the above feature selection problem. Elitism is implemented at each generation by preserving the best string seen up to that generation in a location outside the population. Thus on termination, this location contains the best feature combination.

Table XX. Features Identified by GA-Based Approach (ME Classifier)

Language	Features
Bengali	$C[-2,+2]$, Pre_3, Suf_3, POS_i, digitDot, digitSlash and digitHyphen, t_{i-1}, Sem_i
Hindi	$C[-1,+1]$, Suf_4, Pre_4, POS_i, Infrequent word, digitComma, digitDot and digitSlash, t_{i-1}, Sem_i
Telegu	$C[-1,+1]$, Suf_3, Pre_4, POS_i, digitDot and digitSlash, t_{i_1}, Sem_i

Table XXI. Results of Feature Selection Against Classifier Ensemble Formed Using Only ME-Based Classifiers (We Report in Percentages)

Language	Model	recall	precision	F-measure
Bengali	*Baseline* based on feature selection	86.32	89.02	87.65
Bengali	GA-based classifier ensemble	87.51	89.67	88.57
Hindi	*Baseline* based on feature selection	86.01	90.92	88.40
Hindi	GA-based classifier ensemble	87.52	92.62	90.00
Telugu	*Baseline* based on feature selection	73.80	86.58	79.68
Telugu	GA-based classifier ensemble	75.01	87.95	80.97

7.4 Experimental Results and Discussions

The feature selection technique is evaluated for Bengali, Hindi, and Telugu. Here we have selected best feature combination for ME classifier. But the approach is very general and it can be used with any other classifiers like CRF, SVM etc. We set the following parameter values for GA: population size = 100, number of generations = 50, probability of mutation = 0.2 and probability of crossover = 0.9.

Here, the following features are considered: Context of size five (previous two, current, and next two) or three (previous, current, and next) words, prefixes of length up to three or four characters (three or four features), suffixes of length up to three or four characters (three or four features), POS information, first word of the sentence, length of the word, infrequent word, position of the word, digitComma, digitPercentage, digitDot, digitSlash, digitHyphen, digitFour, digitTwo, semantic feature, and dynamic NE information.

The GA-based feature selection technique finally selects the features as shown in Table XX for these languages. In the table, meanings of the notations are as follows: $C[-i,+j]$: context spanning from the previous i^{th} word to the next j^{th} word with the current token at position 0; Pre_i and Suf_i: prefixes and suffixes of character sequences up to i of the current word, respectively; t_i: output class of the current token; and SeM_i: Semantic feature of the current token.

The recall, precision, and F-measure values of the best individual classifier trained using the feature set identified by the GA-based approach are 86.32%, 89.02%, and 87.65%, respectively for Bengali. Thereafter, we execute the feature selection algorithm for Hindi and Telugu datasets. Evaluation results yield the recall, precision, and F-measure values of 86.01%, 90.92%, and 88.40%, respectively for Hindi and 73.80%, 86.58%, and 79.68%, respectively for Telugu. Overall results of the feature selection technique and the GA-based classifier ensemble are presented in Table XXI. For each of the languages, results show that GA based classifier ensemble performs superior compared to the baseline model developed using the GA-based feature selection approach. But, it is also to be noted that the GA-based feature selection yields better performance in comparison to the heuristics based feature selection of ME model for each of the languages. The GA-based feature selection technique shows the increments of 1.11, 1.99, and 3.04 percentage F-measure points (c.f. M_6 in Table IV, M_6 in Table VII, and M_{13} in Table X) over the best individual ME-based classifiers for Bengali, Hindi, and Telugu, respectively.

8. CONCLUSION AND FUTURE WORKS

In this article, we proposed a novel technique of classifier ensemble for NER. We have used GA to determine the weights of votes for different output classes in each classifier. We hypothesized and experimentally shown that instead of eliminating some classifiers completely, it is better to quantify the amount of votes per classifier for each output class. The proposed approach encodes the weights in its chromosome. Adaptive mutation and crossover probabilities have been used to accelerate the convergence of GA. The overall *F*-measure value of three-fold cross validation on the training data attained by the weighted vote-based classifier ensemble encoded in a particular chromosome has been used as its objective value. We have used ME, CRF, and SVM frameworks as the base classifiers to generate the several different versions of them by varying the available features and/or feature templates. One most interesting and important characteristic of our system is that it makes use of only language independent features that can be easily derived for almost all the languages without any knowledge of the languages a priori. Thus, our proposed algorithm can be very easily replicated for any language. We evaluated our proposed technique for four leading Indian languages, namely Bengali, Hindi, Telugu, and Oriya, which are all resource-constrained in nature. Experiments show the effectiveness of our proposed approach with the promising results for all the four languages. We also evaluated the approach with the benchmark English datasets of CoNLL-2003 shared task, and it showed the performance comparable to the existing state-of-the-art systems. Results also show that the performance attained by the GA-based ensemble technique is better than the best individual classifier, three baseline ensemble techniques and some existing state-of-the art classifier ensemble techniques. We also showed the effects of ensemble size and training set size on the overall system performance.

In part of the article, we have also developed a feature selection technique using GA. This is used as a baseline that showed inferior performance compared to the proposed classifier ensemble.

In this article, we have used GA to optimize only one classification quality measure to determine the best classifier ensemble. But sometimes only a single measure may not always capture the quality of an ensembling reliably. For a good ensemble, sometimes it may be necessary to optimize more than one classification quality measures simultaneously. Thus it would be good to use multiobjective optimization (MOO) techniques to solve the classifier ensemble problem. In the future we will use some MOO-based techniques to solve the problem of weighted vote-based classifier ensemble.

REFERENCES

Alfonseca, E. and Manandhar, S. 1999. An unsupervised method for general named entity recognition and automated concept discovery. In *Proceedings of the 16th National Conference on Artificial Intelligence and the Eleventh Conference on Innovative Applications of Artificial Intelligence (AAAI'99/IAAI'99)*. 474–479.

Anderson, T. W. and Scolve, S. 1978. *Introduction to the Statistical Analysis of Data*. Houghton Mifflin.

Aone, C., Halverson, L., Hampton, T., and Ramos-Santacruz, M. 1998. SRA: Description of the IE2 system used for MUC-7. In *Proceedings of the Message Understanding Conference (MUC'98)*.

Babych, B. and Hartley, A. 2003. Improving machine translation quality with automatic named entity recognition. In *Proceedings of the Conference on the European Chapter of the Association for Computational Linguistics Workshop on Machine Translation and Other Language Technology Tools (EACL'03)*. 1–8.

Bennet, S. W., Aone, C., and Lovell, C. 1997. Learning to tag multilingual texts through observation. In *Proceedings of Empirical Methods of Natural Language Processing (EMNLP'97)*. 109–116.

Bikel, D. M., Schwartz, R. L., and Weischedel, R. M. 1999. An algorithm that learns whats in a name. *Mach. Learn. 34*, 1-3, 211–231.

BORTHWICK, A. 1999. Maximum entropy approach to named entity recognition. Ph.D. thesis, New York University.

BORTHWICK, A., STERLING, J., AGICHTEIN, E., AND GRISHMAN, R. 1998. NYU: Description of the MENE named entity system as used in MUC-7. In *Proceedings of the Machine Understanding Conference (MUC'98)*.

BREIMAN, L. 1996. Bagging predictors. *Mach. Learn. 24*, 2, 123–140.

CARREARS, X., MARQUEZ, L., AND PADRO, L. 2002. Named entity recognition using AdaBoost. In *Proceedings of the Conference on Natural Language Learning (CoNLL'02)*. 167–170.

CHERKAUER, K. 1996. Human expert-level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Working Notes of the AAAI Workshop on Integrating Multiple Learned Models (AAAI'96)*. 15–21.

CHIEU, H. L. AND NG, H. T. 2003. Named entity recognition with a maximum entropy approach. In *Proceedings of the Conference on Natural Language Learning (CoNLL'03)*. 160–163.

COLLINS, M. AND SINGER, Y. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP'99)*.

DARROCH, J. AND RATCLIFF, D. 1972. Generalized iterative scaling for log-linear models. *Ann. Math. Statist. 43*. 1470–1480.

DIETTERICH, T. G. 2002. Ensemble methods in machine learning. In *Proceedings of the 1st International Workshop on Multiple Classifiers Systems*. J. Kittler and F. Roli Eds., Springer.

DIETTERICH, T. G. AND BAKIRI, G. 1995. Solving multiclass learning problems via error correcting output codes. *J. Artific. Intell. Res. 2*, 263–286.

EKBAL, A. AND BANDYOPADHYAY, S. 2007. Lexical pattern learning from corpus data for named entity recognition. In *Proceedings of the 5th International Conference on Natural Language Processing (ICON'07)*. 123–128.

EKBAL, A. AND BANDYOPADHYAY, S. 2008a. Bengali named entity recognition using support vector machine. In *Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages, 3rd International Joint Conference on Natural Languge Processing (NER-IJCNLP'08)*. 51–58.

EKBAL, A. AND BANDYOPADHYAY, S. 2008b. Web-based Bengali news corpus for lexicon development and POS tagging. *POLIBITS, 37*, 20–29.

EKBAL, A. AND BANDYOPADHYAY, S. 2008c. A Web-based Bengali news corpus for named entity recognition. *Lang. Resour. Eval. 42*, 2, 173–182.

EKBAL, A. AND BANDYOPADHYAY, S. 2009. Voted NER system using appropriate unlabeled data. In *Proceedings of the Named Entities Workshop: Shared Task on Transliteration (NEWS'09)*. 202–210.

EKBAL, A., NASKAR, S., AND BANDYOPADHYAY, S. 2007. Named entity recognition and transliteration in Bengali. *Lingvisticae Investigationes J. 30*, 1 (Named Entities: Recognition, Classification and Use Special Issue), 95–114.

EKBAL, A., HAQUE, R., AND BANDYOPADHYAY, S. 2008. Named entity recognition in Bengali: A conditional random field approach. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP'08)*. 589–594.

EKBAL, A. AND SAHA, S. 2010. Weighted vote-based classifier ensemble selection using genetic algorithm for named entity recognition. In *Proceedings of the Conference on Natural Languages in Databases (NLDB'10)*. 256-267.

ETZIONI, O., CAFARRELLA, M., DOWNEY, D., POPESCU, A. M., SHAKED, T., SODERLAND, S., WELD, D. S., AND YATES, A. 2005. Unsupervised named entity extraction from the Web: An experimental study. *Artific. Intell. 165*. 91–134.

FLORIAN, R., ITTYCHERIAH, A., JING, H., AND ZHANG, T. 2003. Named entity recognition through classifier combination. In *Proceedings of the 7th Conference on Natural Language Learning (HLT-NAACL'03)*.

FREUND, Y. AND SCHAPIRE, R. 1995a. A decision-theoretic generalization of online learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory (ECCL'95)*. 23–37.

FREUND, Y. AND SCHAPIRE, R. E. 1995b. A decision-theoretic generalization of online learning and an application to boosting. In *Proceedings of the 2nd European Conference on Computational Learning Theory (ECCL'95)*. 23–37.

GOLDBERG, D. E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, New York.

HOLLAND, J. H. 1975. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press: AnnArbor.

HUMPHREYS, K., GAIZAUSKAS, R., AZZAM, S., HUYCK, C., MITCHELL, B., CUNNIGHAM, H., AND WILKS, Y. 1998. University of Sheffield: Description of the LaSIE-II System as Used for MUC-7. In *Proceedings of the Message Understanding Conference (MUC'98)*.

JOACHIMS, T. 1999. *Making Large Scale SVM Learning Practical*. MIT Press: Cambridge, MA, 169–184.

KLEIN, D., SMARR, H. N., AND MANNING, D. 2003. Named entity recognition with character-level models. In *Proceedings of the Conference on Natural Language Learning (CoNLL'03)*. 188–191.

KOLEN, J. F. AND POLLACK, J. B. 1991. Back propagation is sensitive to initial conditions. *Adv. Neural Inf. Proc. Syst.*. 860–867.

LAFFERTY, J. D., MCCALLUM, A., AND PEREIRA, F. C. N. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML'01)*. 282–289.

LI, W. AND MCCALLUM, A. 2004. Rapid development of Hindi named entity recognition using conditional random fields and feature induction. *ACM Trans. Asian Lang. Inform. Process. 2*, 3, 290–294.

LIN, D. AND WU, X. 2009. Phrase clustering for discriminative learning. In *Proceedings of 47th Annual Meeting of the Association for Computational Learning (ACL'09)*. 1030–1038.

MANDL, T. AND WOMSER-HACKER, C. 2005. The effect of named entities on effectiveness in cross-language information retrieval evaluation. In *Proceedings of the ACM Symposium on Applied Computing (SAC'05)*. 1059–1064.

MCCALLUM, A. AND LI, W. 2003. Early results for named entity recognition with conditional random fields, feature induction, and Web-enhanced lexicons. In *Proceedings of the Conference on Natural Language Learning (CoNLL'03)*. 188–191.

MIKHEEV, A., GROVER, C., AND MOENS, M. 1998. Description of the LTG system used for MUC-7. In *Proceedings of the Message Understanding Conference (MUC'98)*.

MIKHEEV, A., GROVER, C., AND MOENS, M. 1999. Named Entity Recognition without Gazeteers. In *Proceedings of the Conference on the European Chapter of the Association for Computational Linguistics (EACL'03)*. 1–8.

MILLER, S., CRYSTAL, M., FOX, H., RAMSHAW, L., SCHAWARTZ, R., STONE, R., WEISCHEDEL, R., AND THE ANNOTATION GROUP. 1998. BBN: Description of the SIFT System as Used for MUC-7. In *Proceedings of the Message Understanding Conference (MUC'98)*.

NOBATA, C., SEKINE, S., ISAHARA, H., AND GRISHMAN, R. 2002. Summarization system integrated with named entity tagging and IE pattern discovery. In *Proceedings of 3rd International Conference on Language Resources and Evaluation (LREC'02)*.

PASCA, M., LIN, D., BIGHAM, J., LIFCHITS, A., AND JAIN, A. 2006. Organizing and searching the World Wide Web of facts - Step one: The one-million fact extraction challenge. In *Proceedings of National Conference on Artificial Intelligence (AAAI'06)*.

PATEL, A., RAMAKRISHNAN, G., AND BHATTACHARYA, P. 2009. Relational learning assisted construction of rule base for Indian language NER. In *Proceedings of the 7th International Conference on Natural Language Processing (ICON'09)*.

PIZZATO, L. A., MOLLA, D., AND PARIS, C. 2006. Pseudo relevance feedback using named entities for question answering. In *Proceedings of the Australian Language Technology Workshop (ALTW'06)*. 89–90.

RILOFF, E. AND JONES, R. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99)*. 474–479.

SAHA, S., SARKAR, S., AND MITRA, P. 2008. A hybrid feature set based maximum entropy Hindi named entity recognition. In *Proceedings of the 3rd International Joint Conference in Natural Langauge Processing (IJCNLP'08)*. 343–350.

SEKINE, S. 1998. Description of the Japanese NE system used for MET-2. In *Proceedings of the Message Understanding Conference (MUC'98)*.

SEUNG, H. S., OPPER, M., AND SOMPOLINSKY, H. 1992. Query by committee. In *Proceedings of the ACM Workshop on Computational Learning Theory (CLT'92)*.

SHA, F. AND PEREIRA, F. 2003. Shallow parsing with conditional random fields. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL'03)*. 134–141.

SHINYAMA, Y. AND SEKINE, S. 2004. Named entity discovery using comparable news articles. In *Proceedings of the International Conference on Computational Linguistics (COLING'04)*. 848–855.

Shishtla, P. M., Pingali, P., and Varma, V. 2008. A character n-gram based approach for improved recall in Indian language NER. In *Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages (IJCNLP'08)*. 101–108.

Srihari, R., Niu, C., and Li, W. 2002. A hybrid approach for named entity and sub-type tagging. In *Proceedings of 6th Conference on Applied Natural Language Processing (ANLP'02)*. 247–254.

Srikanth, P. and Murthy, K. N. 2008. Named entity recognition for Telugu. In *Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages (IJCNLP'08)*. 41–50.

Srinivas, M. and Patnaik, L. M. 1994. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Trans. Syst. Man Cybern. 24*, 4, 656–667.

Suzuki, J. and Isozaki, H. 2008. Semi-supervised sequential labeling and segmentation using gigaword scale unlabeled data. In *Proceedings of the Human Language Technology Conference (ACL/HLT'08)*. 665–673.

Taira, H. and Haruno, M. 1999. Feature selection in SVM text categorization. In *Proceedings of National Conference on Artificial Intelligence (AAAI'99)*.

Tjong Kim Sang, E. F. and De Meulder, F. 2003. Introduction to the shared task: Language independent named entity recognition. In *Proceedings of the 7th Conference on Natural Language Learning (HLT-NAACL'03)*. 142–147.

Vapnik, V. N. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag Berlin, Germany.

Vijayakrishna, R. and Sobha, L. 2008. Domain focused named entity recognizer for Tamil using conditional random fields. In *Proceedings of the Workshop on Named Entity Recognition for South and South East Asian Languages (IJCNLP'08)*. 93–100.

Wolpert, D. 1992. Stacked generalization. *Neural Netw. 5*, 241–259.

Wu, D., Ngai, G., and Carput, M. 2003. A stacked, voted, stacked model for named entity recognition. In *Proceedings of the Conference on Natural Language Learning (CoNLL'03)*.

Yangarber, R., Lin, W., and Grishman, R. 2002. Unsupervised learning of generalized names. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'02)*. 1–7.

Yu, X. 2007. Chinese named entity recognition with cascaded hybrid model. In *Proceedings of Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics (NAACL-HLT'07)*. 197–200.

Received May 2010; revised September 2010, December 2010, January 2011; accepted January 2011